Final Report

ESMART Subtask 3.7: Design Support for Tooling Optimization

Award Number DE-FC36-04GO14230
Co-Op Agreement No. 2005-307
OSURF Project Number 746201

R. Allen Miller, Principal Investigator
Phone: 614-581-1754
Email: Miller.6@osu.edu
Ohio State University
210 Baker Systems Engineering
1971 Neil Ave.
Columbus, OH 43210

Contributors:
Khalil Kabiri- Bamoradian, Research Engineer
Dongtao Wang, PhD Student
Hongyu Xu, PhD Student
Li Shen, MS Student

September 23, 2011

Acknowledgement/Disclaimer

Acknowledgement: This report is based upon work supported by the U S. Department of Energy under Award No. DOE award DE-FC36-04GO14230.

Disclaimer: Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the Department of Energy.

Proprietary Data Notice: This report does not contain any proprietary data.
Document Availability: Reports are available free via the U.S. Department of Energy (DOE) Information Bridge Website: http://www.osti.gov/bridge

Reports are available to DOE employees, DOE contractors, Energy Technology Data Exchange (ETDE) representatives, and Informational Nuclear Information System (INIS) representatives from the following source:

Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831
Tel: (865) 576-8401
FAX: (865) 576-5728
E-mail: reports@osti.gov
Website: http://www.osti.gov/contract.html

Table of Contents

List of Acronyms

AMC	American Metalcasting Consortium
CMC	Cast Metals Coalition
DOF	Degree of freedom
NADCA	North American Die Casting Association
ODE	Ordinary differential equation
STL	Stereo Lithography

List of Figures

List of Tables

List of Appendices

Executive Summary

High pressure die casting is an intrinsically efficient net shape process and improvements in energy efficiency are strongly dependent on design and process improvements that reduce scrap rates so that more of the total consumed energy goes into acceptable, usable castings. Computer simulation has become widely used within the industry but use is not universal. Further, many key design decisions must be made before the simulation can be run and expense in terms of money and time often limits the number of decision iterations that can be explored. This work continues several years of work creating simple, very fast, design tools that can assist with the early stage design decisions so that the benefits of simulation can be maximized and, more importantly, so that the chances of first shot success are maximized. First shot success and better running processes contributes to less scrap and significantly better energy utilization by the process.

Part of the work improved the fill pattern visualization algorithm used in the CastView casting design visualization program. The algorithm was also extended to address slower filling casting processes such as squeeze casting, permanent mold casting, and sand casting. The dominant term(s) for flow behavior for each process were defined from a dimensionless form of the Navier-Stokes equations. Based on this dimensional analysis, the fill pattern algorithm originally developed for high pressure die casting was modified for slow fill processes by establishing an effective flow direction using the vector sum of the dominant forces, typically inertial and gravity forces. Test results using the slow fill algorithms were compared with those obtained from numerical simulations, historical data and experiments. The comparison shows very good agreement and computational time of a few minutes.

The second part of the work documents a practical way to model the large dynamic spike in the metal pressure at the end of fill in high pressure die casting. The pressure spike is due to the rapid deceleration of the liquid metal, plunger, and hydraulic oil at the end of fill. High pressure die casting injects the molten metal with a hydraulic piston that is moving, essentially, as fast as possible during fill. A considerable quantity of mass moving at high speed results in a large amount of kinetic energy that must be dissipated at the end of fill. There is limited, if any, deceleration which means the energy is dissipated by crashing the piston and molten metal into the die and machine. The result is a very rapid deceleration and a corresponding rapid rise in the metal pressure. This spike plays a key role in die flashing. The fact that the cavity pressure and plunger speed are tightly coupled makes prediction of transient cavity pressure extremely difficult using current fluid flow simulation approaches that depend on a predefined shot speed profile that is not coupled to the forces present in the system. The model that was developed can predict the transient cavity pressure at the end of fill and provide insight into the influence of the shot profiles on the cavity pressure spike. The model also can be used as an input for die distortion modeling. The model is a multi-degree of freedom lumped-parameter model that represents the injection system and die casting machine in terms of a simple spring-mass-damper elements that addresses the coupling between plunger speed and pressure and predicts the transient cavity pressure during cavity filling in a computationally inexpensive way. Different shot plunger deceleration profiles were studied and useful insights into the die responses were achieved. As expected, the results show quantitatively that a decelerated plunger shot profile can dramatically decrease the impact pressure spike. In addition, the problems associated with attempting to use the familiar numerical fill simulations to predict pressure at the end of cavity fill are also discussed and illustrated. The lumped parameter model developed in this work is one way around these problems.

The last component of the work addresses the use of an approximate equilibrium die temperature model for cooling system and casting cycle optimization. The technique was demonstrated with a study carried out in cooperation with an industry collaborator and provides a dramatic demonstration of the benefits of optimizing the thermal behavior of dies. The die in question is a small four cavity die producing zinc parts on a small machine. Operations with the original die design resulted in frequent tie bolt failure due to low cycle fatigue. The root cause of the failure was due to non-uniform thermal growth of the die. By adjusting the orientation of the parts in the die, modifying the cooling line layout, and the cooling line operation, the non-uniform growth was essentially eliminated and the cycle time was reduced by 40% at the same time. The results provide a strong argument for optimization of the thermal performance of die casting dies, or at least treating die layout and cooling system performance as important die design issues.

This new technology was predicted to result in an average energy savings of 1.83 trillion BTU’s/year over a 10 year period. Current (2011) annual energy saving estimates over a ten year period, based on commercial introduction in 2012, a market penetration of 30% by 2015 is 1.89 trillion BTU’s/year by 2022.

Along with these energy savings, reduction of scrap and improvement in yield will result in a reduction of the environmental emissions associated with the melting and pouring of the metal which will be saved as a result of this technology. The average annual estimate of CO_2 reduction per year through 2022 is 0.037 Million Metric Tons of Carbon Equivalent (MM TCE).

.

1. Introduction

Die casting is a permanent mold near net shape process in which metal is injected into the mold cavity at high speed and pressure. It is typically used in situations where high part volumes and production rates are required. It is also a technologically complex thermal process where energy utilization is very strongly affected by the part and die designs and by how well the die casting machine is matched to the requirements of the part and die. The process is capable of producing geometrically complex parts to precise tolerances with good surface finish, but the process can be susceptible to thermal, flow and dimensional defects that reduce the overall process yield with a corresponding increase in energy usage. Practitioners all agree that the best path to reducing the chance of defects includes ensuring that the part geometry is as castable as possible and the fill and thermal aspects of the part/die/machine system are carefully engineered. The basic purpose of this work is to develop methods that support part, die and process design tasks with minimal computational requirements.

The Center for Die Casting at The Ohio State University developed cavity fill visualization techniques that were implemented in the CastView software program distributed by the North American Die Casting Association (NADCA). One of the objectives of the work reported here is to improve the reasoning algorithms reducing certain inaccuracies that occur in the fill patterns computed for high pressure die casting. In addition, the reasoning methods that were successfully applied to die casting will be extended to other casting processes including squeeze, semi-solid, permanent mold casting and sand casting. This will provide the utility of the visualization approach to additional important casting processes thus improving part and mold designs and ultimately reducing scrap and wasted energy.

The second area of work will develop very simple models of the dynamics of the impact spike that occurs at the end of cavity fill in high pressure die casting as the plunger that is pushing the metal into the cavity rapidly comes to a stop once the cavity is filled. This is a very difficult problem to address with current fluid and structural finite element modeling but relatively simple to address with dynamic models based on point mass approximations and system force balances. The models help design better shot profiles which will help eliminate die flashing and scrap due to parts failing to meet dimensional tolerances. Also, a die that flashes generally results in an inconsistent operating cycle since the operator periodically has to clean the parting surface of the die. Every interruption of the normal cycle allows the die to cool below its steady state operating conditions and requires a few cycles to get back to normal conditions. Often the parts produced as the die returns to normal conditions must be scrapped making flash a very serious problem in terms of energy utilization. The relatively simple model that is developed provides a very useful tool to minimize this problem.

The last area of work addresses improving/optimizing the thermal balance in die casting dies. A previous Cast Metals Coalition (CMC) project successfully developed techniques to compute the die equilibrium temperature (spatial distribution of the average cycle temperature once the process reaches quasi steady-state). The part ejection temperature is also computed. These are accomplished with very short run times and minimum setup and without requiring computation of numerous casting cycles as is normally required with computer simulation. These techniques provide the opportunity to optimize the die cooling and cycle in any process that uses a permanent mold and with modest of computing power.

The technologies developed in this work are not complex and that is the power. All are supported with relatively simple computer applications and can be effectively used by essentially any caster. The commercialization plan for this work takes advantage of these facts and the technology will become part of NADCA's Productivity Enhancement Suite. New versions of the CastView program incorporating the fill and thermal work will soon be released. The impact spike work will be converted to a simple application that can be run from the web or downloaded and will be incorporated as a module in the PQ^2 that is distributed by NADCA.

2. Background

The purpose of this section is to provide a summary of related previous work and to outline the objectives of the process.

2.1 Summary of Previous Work

2.1.1 Cavity Fill Simulation and Visualization

With the support of CMC and American Metalcasting Consortium (AMC) funding over the past several years, the Center for Die Casting at The Ohio State University has developed part design evaluation tools based primarily on qualitative reasoning and implemented them in a software system called *CastView* that is now distributed by NADCA. The methods behind the program were constructed with the explicit objective of providing a very fast, conservative, screening tool that could provide up front design information in a matter of a few minutes thereby maximizing the number of 'what ifs' that a designer could explore in the time available. The data provided are not full details, but general indicators of potential problems such as hot spots, shrinkage, difficult fill conditions, difficult vent conditions and the corresponding trapped air problems that go along with venting, die steel conditions that result in high maintenance costs, and some ejectability issues. Since the data are visual and can be obtained very quickly, the program greatly helps users clearly communicate with their customers about potential problems and hopefully fix them even before proceeding with simulation work and detailed die design.

At present, the prevailing method to estimate a fill pattern is numerical simulation. A simulation typically sets up equations based on mass conservation, momentum conservation and energy conservation then solves the equation system [1-3]. Due to the large number of equations, equation solving is usually a time consuming task. Furthermore, the simulation systems usually require the user to have considerable experience and good understanding of fluid dynamics. These reasons limit the number of "what-ifs" that can be performed and also limit numerical simulation's utility in the early process design stage.

One of the alternative methods is qualitative reasoning for fill pattern prediction. This method is based on the geometric characteristics of cavity so it only requires the input of part and gate geometry and does not require fluid dynamics background of the user. Moreover, there are no complex equations to be solved so the analysis can be finished quickly, which is very desirable for the early stage design. The pioneers using geometric reasoning for die casting process analysis are Uicker and his colleagues at the University of Wisconsin [4]. The CastView research group at The Ohio State University developed a qualitative reasoning algorithm for fill pattern prediction [5-8]. The input is part geometry in STL or stereo lithography format (a triangular faceted representation) on which the user can define the gates. The STL part is first discretized into a voxel model (a cubic box representation) and the fill pattern is determined based on computational geometric reasoning with typical run time of a few minutes.

The application of fill pattern analysis in CastView has proved that this is a simple and efficient way to predict fill pattern in die casting cavity. Though the accuracy level is not as high as that of numerical simulation and the information provided is incomplete, it provides a quick but reasonable solution for the fill pattern analysis. There were, however, occasional problems of the analysis in some cases. Therefore, it is very necessary to improve the original algorithm to obtain a more accurate result. This report presents the method of improvement and compares the new results with old results and with those of numerical simulation.

2.1.2 Impact Spike

In the high-pressure die-casting process, a plunger pushes molten metal into a die cavity at very high speed. The molten metal, the plunger, plunger rod, hydraulic piston, and some amount of hydraulic oil are in motion at a high speed during cavity fill and rapidly decelerate and stop at the end of fill. The sudden deceleration of all these moving masses can create a very high dynamic pressure – impact pressure, in the die cavity depending on the masses of the system and deceleration time. The role of cavity pressure and the effects of impact pressure developed at the end of fill have been documented in the literature, as discussed below.

The pressure spike in the cavity is caused by the force needed to decelerate the moving masses. The cavity pressure peak is responsible for large deflections of the die which may lead to flashing [9]. The effect of high pressures on casting quality was investigated and the insights indicated that high impact pressure does not always improve the casting quality as high intensification pressure. The high impact pressure or high plunger velocity makes flashing more likely [16]. J. R. Mickowski et al [14] stated that the impact pressure should be controlled such that the plunger velocity and final pressure are retained. They also developed a real-time closed-loop shot control system to control the impact pressure. In this system, a rapid response valve was added to the hydraulic cylinder so that an extremely rapid deceleration in a small final fraction of the plunger stroke can be applied, by which overall cavity fill time at least as short as in the un-controlled conventional process was maintained [14]. Peter Olmsted developed a very innovative idea that uses the overflows to cushion the impact pressure. By adjusting the overflow volume and the entry size to overflows, the impact pressure can be controlled in a very robust way [10].

Limited experimental research in cavity pressure measurements has been done. G. Savage et al [16] put a load cell inside the plunger rod coupling to measure shot rod force during injection process. The backpressure of molten metal and intensification pressure were derived from the shot rod force and plunger tip area. K. Tong et al [18] presented a measurement method in thin-wall magnesium die casting experiments. Quartz force sensors were placed behind the ejector pins, and the metal pressure was calculated from the ejector pin force and its tip area [18]. The methods mentioned above are indirect measurements.

Cavity pressure is very difficult to be accurately measured due to the complexity of die casting process and the very fast speed of response. Computer simulation of the injection process thus becomes a logical choice to predict the transient cavity pressure. However, in most fluid flow modeling, the pressure is computed from a prescribed plunger velocity, which is an uncoupled approach (velocity is independent of pressure). The general purpose CFD software FLOW-3D[1] has the capability of modeling fully coupled motion of solid bodies in fluids by applying FAVOR^{TM} technique to general moving objects (GMO), which can be used to simulate die cavity filling by modeling the plunger and allows for position dependent hydraulic driving force [21]. However, numerical fluid volume loses that occur during this computation cause trouble in predicting the pressure. The molten metal is always assumed as either incompressible or slightly compressible fluid; therefore the pressure impact is very sensitive to the volume change of molten metal due to its very large bulk modulus. Pressure prediction may be meaningless if the numerical volume of molten metal changes obviously between time steps.

During the plunger deceleration phase, the cavity pressure and plunger movement are highly coupled due to the fact that the interaction between the hydraulic pressure and metal pressure forces the plunger into oscillation, which decays to a full stop. Accurate pressure results especially the impact pressure spike at the end of filling cannot be obtained using an uncoupled simulation [20].

2.1.3 Thermal and Cycle Optimization

Appropriate thermal balance is a primary key to ensuring quality die castings and prolonging the die life. It is not surprising that thermal simulation was among the first areas for which simulation tools were developed [19] and today thermal analysis is widely applied in industry.

The typical procedure for numerical simulation is to build CAD models for part and dies – input thermal property data and cycle parameters – run simulation – view analysis results. The analysis results include the temporal and spatial temperature pattern of the part and dies. Thus the user can read temperature at any location and at any time from the results. The information is quite complete and gives a good depiction of the thermal performance of the die. However, one disadvantage of transient solutions is that to obtain the thermal profile at quasi steady state literally hundreds of cycles are needed since the start-up temperature distribution is usually not close to the steady state and the die temperature will continue to change from cycle to cycle until the quasi-equilibrium is achieved. The multiple cycle simulation is time consuming and although it provides very complete information, some of the information is beyond that needed for the cycle and die cooling design since cycle and cooling design only requires the overall temperature distribution and part ejection temperature at steady state. Extra information is not bad, however the extra time required may limit exploration of "what-ifs." Thus it may be useful to have a tool for thermal analysis that provides the needed information and runs very quickly supporting interactive redesign and exploration of as many "what-ifs" as possible. The tradeoff is completeness of information with the time required for the analysis.

[1] FLOW-3D is a registered trademark of Flow Science Inc.

In quasi steady state, the temperature of a given point in the die may vary throughout the casting cycle depending on the distance from the cavity surface but the point returns to the same value at the same relative time in each cycle. In this work the equilibrium temperature at such a point is defined as the time average temperature over a cycle after the process reaches quasi steady state which is illustrated using Figure 1 and Figure 2. Considering an arbitrary hypothetical point in the die, its temperature change over time is shown in Figure 1. After a sufficient number of start-up cycles, cycle to cycle change is small and the process reaches the quasi steady state as illustrated in Figure 2. The equilibrium and the transient temperatures for a die in quasi-equilibrium will be very close except for points relatively close to the cavity surface. Points on the cavity surface vary considerably in temperature over the cycle, but points in the interior of the die body do not. Note that this is time average and not a spatial average so the temperature still varies spatially. This equilibrium temperature is helpful for cycle and cooling design, especially at conceptual design stage. A part of the research is to develop and implement a quick approach to compute the equilibrium temperature of die and part.

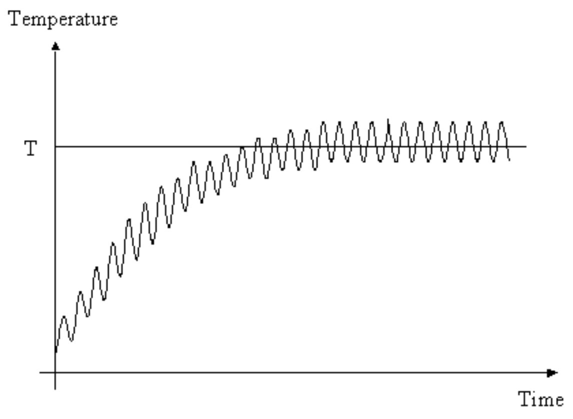


Figure 1 Hypothetical temperature change of a point in the die over time

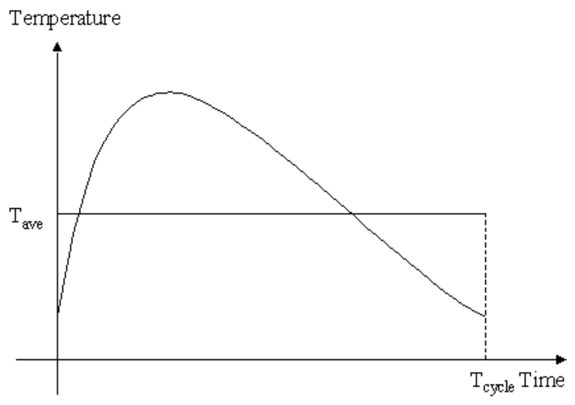


Figure 2 Temperature change of point over a cycle in equilibrium

2.2 Objectives

As described in the introduction, the objective of this work, in general, is to develop simple design tools that assist casting, die, and process designers to establish effective designs. Of interest are methods to quickly assess the fill pattern of a casting in order to properly size and place gates and overflows; a model to capture the dynamic pressure spike that occurs at the end of cavity filling in order to design better shot profiles; and demonstrate the use of equilibrium temperature calculations to optimize die cooling and cycle time. The emphasis in all cases is simplicity, speed and utility. The methods developed must be intuitive, have a low computational burden, and produce results suitable to the design task.

The specific objectives are as follows:

1. Improve the algorithms used for the qualitative reasoning approach to fill visualization of high pressure die casting.
2. Extend the qualitative reasoning based fill algorithms to include slow-fill processes.
3. Develop an approximate model of the dynamic impact spike that occurs in cavity pressure at the end of fill in high pressure die casting.
4. Utilize the efficient equilibrium die temperature distribution procedures and develop procedures suitable to optimize/improve die temperature balance and process cycle time.

2.3 Tasks and Approach

The tasks designed to meet the project objectives are:

1. **Extend Fill Pattern Visualization** - The fill pattern visualization algorithm that is currently used in the CastView program is based on the fact that cavity fill in high pressure die casting is primarily inertia driven. The molten metal is forced through the gate at high velocity and hence with very high momentum. Viscosity and gravity have relatively little effect on the fill pattern under these conditions. Once the material hits obstructions other factors start to play a larger role. The task has two main subtasks that address limitations and extend the technology to additional processes. The two subtasks and the basic approach to each are:

   A. Eliminate inefficiencies and certain difficulties with the basic inertia-controlled filling of high pressure die castings by improving the geometric calculations used to determine how flow obstructions modify the flow pattern.

   B. Extend visualization techniques to slower fill processes by modifying the qualitative reasoning techniques to address visualization of viscosity dominated or gravity dominated or mixed mode flows. The characteristics of the flow visualization (momentum, viscosity, inertia) are made adjustable according to the casting process and used to determine the qualitative flow vector.

2. <u>**Impact Pressure Spike Modeling**</u> - The impact pressure, combined with the dynamic responses of the die machine and dies, is an important factor to consider in machine selection and die design. It is also difficult to address with conventional finite element modeling, but is addressable with approximate models that are appropriate for use in design decision making. The objective of this task is to develop a simplified dynamic model of the transient cavity pressure that can be used in machine selection, shot profile design, die distortion modeling, and gating design. The model provides insight about the influence of the shot profiles on the cavity pressure. A basic lumped parameter (point mass) model of the moving masses (hydraulic pistons, oil, and molten metal) will be developed and a force balance linking the pressure on the shot piston with the acceleration of the masses will be constructed. The possibility of coupling the lumped model with fill simulations will be tested. This is accomplished by using the approximate model to calculate the shot velocity based on the simulation's metal pressure calculation at each step of the fill. The lumped model results can be used stand alone as well.
3. <u>**Thermal/Cycle Optimization**</u> - This task builds on the techniques developed in a previous project that efficiently compute an approximate equilibrium die temperature distribution. The heat balance in the die iteratively improved by adjusting the locations/placement of heating or cooling lines. The die configuration, or orientation and placement of the part in the die, can similarly be evaluated and optimized.
4. **Design Wizards** – The original plan called for knowledge about good design from the NADCA design standards was to be used to define the decision sequence and data to be extracted from various castability analyses used for die and process design.

3. Results and Discussion

The results from each task are discussed below

3.1 Fill Pattern Visualization Extensions

A brief summary of the high pressure die casting fill visualization methods and the extensions to apply them to slow fill processes are discussed in this section.

3.1.1 High Pressure Casting Fill Visualization Improvements

The major adjustments and enhancements are described in the following subsections. A paper describing the details of this work is located in Appendix A.

3.1.1.1 Summary of Improvements

The basic qualitative reasoning approach that is used for visualization discretizes the cavity volume into voxels (discrete cubes) that are geometrically similar to a finite difference grid. Flow is visualized by extending the direction of flow established by the gates into the cavity and continuing along that direction until interference. Due to the discrete voxel space, there are limited directions for flow exiting from any given voxel.

In discrete voxel space, an input flow vector is "rounded" to the nearest discrete vector available. For example, in Figure 3, the output vector of voxel “a” must be rounded to available $\overrightarrow{ab}$ or $\overrightarrow{ac}$, which are based on one layer outer voxels (first set). If the second outer layer is counted, the vectors $\overrightarrow{ad}$ or $\overrightarrow{ae}$ are also available (second set). To keep both vector accuracy and a small computation domain, at most there are five sets of vector available (vectors pointing to the fifth outer layer).

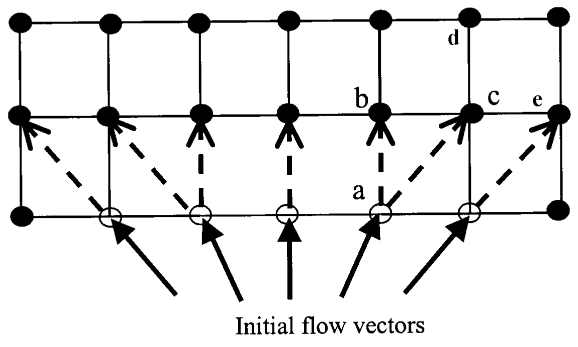


Figure 3 Possible Output Vectors From a Voxel

Problems with the original algorithm may include:

- Speed is not considered - flow has different directions but always the same speed.

- Flow resistance is not considered. Flow resistance depends on the local geometric characteristics on the flow. For example, flow at a thin section generally has larger resistance than at thick section.
- Vector calculation at an obstruction is incomplete. In the original algorithm, when a flow hits the obstruction, it would disperse to all nearby empty voxels.
- Multiple gates not considered. The flow may start cavity filling at one gate earlier than others. These existing problems may cause errors and inconsistencies and they should be improved with the new algorithm.

The basic idea is to correct the problems mentioned earlier but not to increase computation time or memory requirements significantly since the efficiency is a principal goal. The following improvements have been accomplished:

Speed calculation - In the fill pattern algorithm, a list is used to store the flow front voxels. The initial flow front voxels are gate voxels with the initial incoming vector. In each calculation step, the flow front voxels will fill some empty voxels and generate new flow front voxels. This calculation is continued until the whole cavity is filled.

In the new algorithm, speed is included (qualitatively) and each flow front voxel may have a different speed. Both speed and the direction of flow may change during cavity filling. There are three possibilities that cause speed change:

- When the flow hits an obstruction, as the flow vector changes, the speed drops due to energy loss;

- When a voxel is filled from more than one voxel, the speed of outgoing voxel is calculated based on the speed of incoming voxels;

- When flow resistance changes due to local geometry, the flow speed then may change.

Flow resistance - A flow resistance for each flow front voxel is defined based on local geometric characteristics. One example of variable flow resistance is shown in Figure 4. Flow coming from A will generally fill B earlier than fill C since flow resistance at C is larger than that at B.

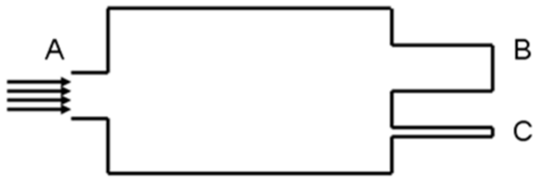


Figure 4 Flow will fill B earlier than fill C

To calculate the flow resistance, we need to consider both the local openness and flow vector at the same time. This can be shown by the example in Figure 5. A flow line is first constructed along flow vector (dashed line in Figure 5) and we count the number of part voxels in the local region at this flow line, denoting the number as *m*. Then a flow plane perpendicular to the flow line is constructed through voxel A, shown as an ellipse in Figure 5. We count the number of part voxel of local region at the flow plane, denoting as *n*. The flow resistance is computed as $r = 1/(m \times n)$. By this definition, we address both local openness and the flow vector. The flow resistance then can be used to compute flow speed.

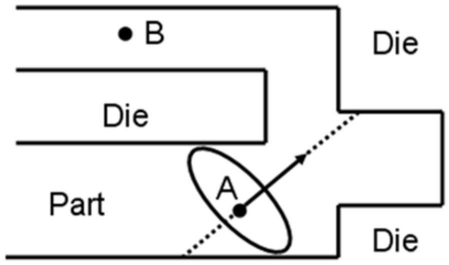


Figure 5 Calculation of flow resistance

Vector change at obstruction - When a flow hits an obstruction, the flow vector must change. In the previous algorithm, the new flow vector was calculated by simply constructing a vector from the current flow front voxel to an empty voxel. With the improvement, this is changed to the sum of original vector and position vector (vector from current voxel to empty voxel). The new direction is not only related to position but also related to original direction, which gives a better way to calculate new direction.

Multiple gates - If a part has multiple gates with different distances to the biscuit and a properly designed runner system, the flow will start to fill cavity simultaneously due to the balanced runner volumes and the large resistance at gates. Changes to the algorithm were made so that the pattern is sensitive to the resistance and not just the distance.

Bias removal - The fill pattern for a symmetric part should also be symmetric; however, this was not always the case. Careful inspection discovered that the problem was caused by calculation sequence. A list is used to store the flow front voxels. At each calculation step, the new flow front voxels are generated based on the sequence of old flow front voxels, i.e. from the front to the end of the list. Since the number of voxel that can be filled in a step is fixed due to mass conservation, there is competition for flow front voxels and voxels at the head of the list will be filled earlier than those at the end. As the calculation continues, there is more and more bias. The solution is to reverse the sequence at each step. At an odd step, the calculation is from the front of flow front list to its end while at an even step, it is from the end to the front.

Efficiency - Special attention was paid to efficiency of the new algorithm. Some of the methods applied to reduce computation:

1. Pre-computed tables of geometry information are used.
2. Real data are rounded to integers.
3. Variables are saved for later use.
4. Data structure and computation are optimized.

With these treatments, the run time increase for the new algorithm is 20% ~ 50% compared to the old.

3.1.1.2 Analysis Results

The algorithm was compared to the numerical simulation results. The first part for test is a simple flat plate. The result from MagmaSoft is shown in Figure 6 while result from the modified visualization algorithm is shown in Figure 7. The results are quite similar. The flow goes into the central region of part and fills the two far corners and then goes back. The two near corners are the last regions to be filled.

The second test part is a finger part that tests the flow resistance and direction change modifications. The numerical simulation result using MagmaSoft is shown in Figure 8. It can be seen that the first finger is filled simultaneously with the second half of part and the second finger is the last region to be filled. The modified algorithm results, shown in Figure 9, are very similar. The first finger is not filled until the second half of part is filled. The second finger is the last region to be filled.

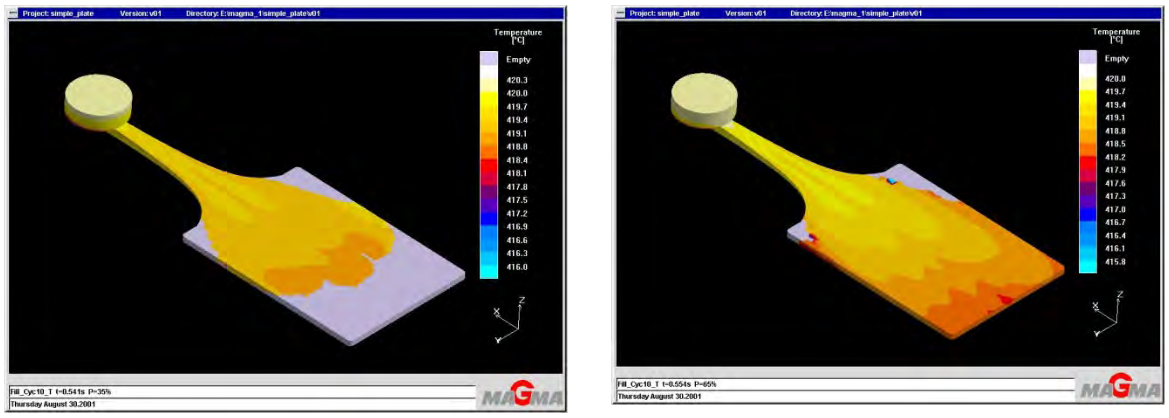


Figure 6 Results from MagmaSoft

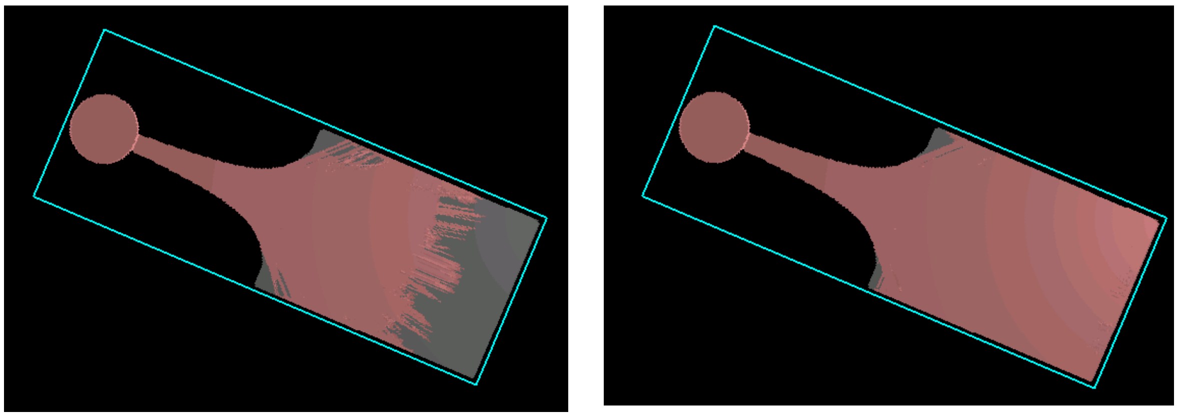


Figure 7 Results from new algorithm

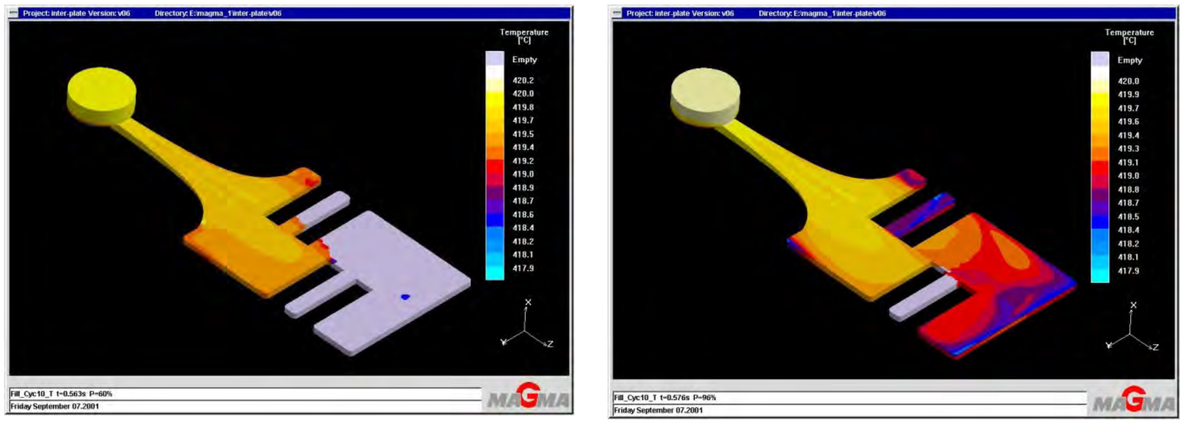


Figure 8 Finger part MagmaSoft results

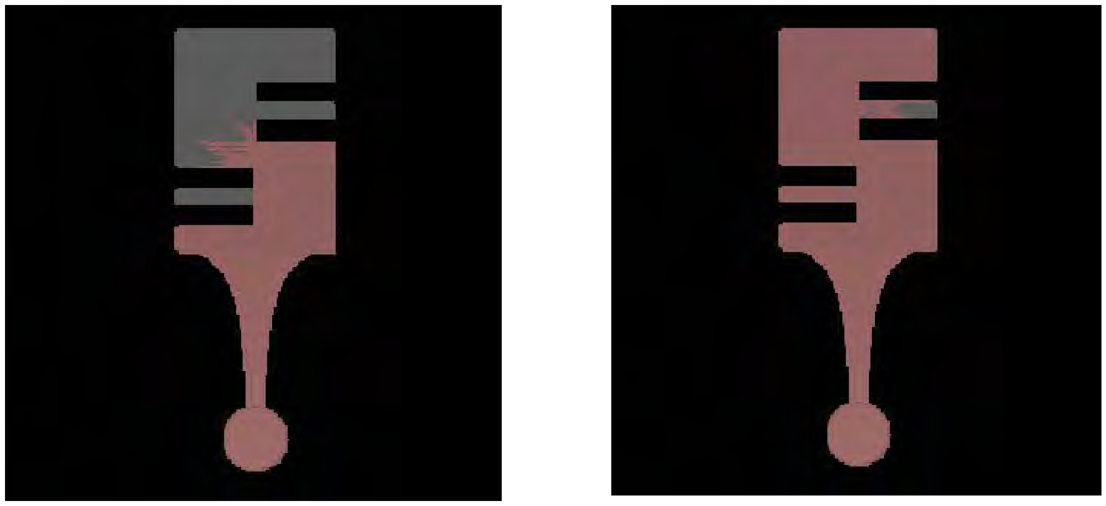


Figure 9 New algorithm, finger part

3.1.2 Extension to Slow Fill Process

Extension of the qualitative reasoning methods to address slow fill processes was a major accomplishment. The details of this work are provided in Appendix B.

A dimensionless form of the Navier-Stokes equation was used to determine the relative importance of inertial, viscous, and gravity forces as a function of the typical parameters associated with each casting process of interest.

High Pressure Die Casting:

Parameters typical of die casting of aluminum alloy are:

$\mu = 3\times10^{-3}\, Pa\ s$ --- Viscosity

$\rho = 2.7\times10^{3}\, kg/m^{3}$ --- Density

$V = 10 m/s$ --- Speed

$L = 0.1 m$ --- Fill distance

$B = 9.8 m/s^{2}$ --- Z component of Gravity

The relative contributions to the Navier-Stokes equations (see Appendix B for details) are

Viscous term: $\frac{\mu V}{L^2}$ = $3\times10^{-3}\times10 / 0.1^2 = 3$.

Inertial term: $\frac{\rho V^2}{L}$ = $2.7\times10^{3}\times10^{2} / 0.1 = 2.7\times10^{6}$.

Gravity term: ρB = $2.7\times10^{3}\times9.8 = 2.6\times10^{4}$.

Clearly, the dominant term is the inertial term. The viscous term is negligible and the gravity term is two orders of magnitude smaller than the inertial term and also negligible.

For gravity casting and squeeze casting, the speed is 0.5 *m/s* and 1 *m/s* respectively and the corresponding data are (assuming the material properties remain the same):

Gravity casting:

Viscous term: $3\times10^{-3}\times0.5 / 0.1^2 = 0.15$.

Inertial term: $2.7\times10^{3}\times0.5^{2} / 0.1 = 6.8\times10^{3}$.

Gravity term: $2.7\times10^{3}\times9.8 = 2.6\times10^{4}$.

Squeeze casting:

Viscous term: $3\times10^{-3}\times1 / 0.1^2 = 0.3$.

Inertial term: $2.7\times10^{3}\times1^{2} / 0.1 = 2.7\times10^{4}$.

Gravity term: $2.7\times10^{3}\times9.8 = 2.6\times10^{4}$.

For gravity casting, the dominant force is gravity but the inertial term is smaller, but still consequential. The viscous term is still negligible. For squeeze casting, both the inertial term and gravity term are essentially equal.

To model the fill pattern of gravity casting and squeeze casting, two component vectors are applied to represent the effects of inertial term and gravity term, namely a momentum vector and a potential vector. The momentum vector depends on the flow velocity while the potential vector depends on the flow location. The combination of these two vectors determines the flow vector. In die casting, momentum vector is the dominant vector since the filling process is under high pressure with high velocity while in gravity casting, the gravity vector is the dominant vector since the velocity is small. If both flows are dominating (for squeeze casting), the actual flow is the sum of both flows. These characteristics are illustrated in Figure 10.

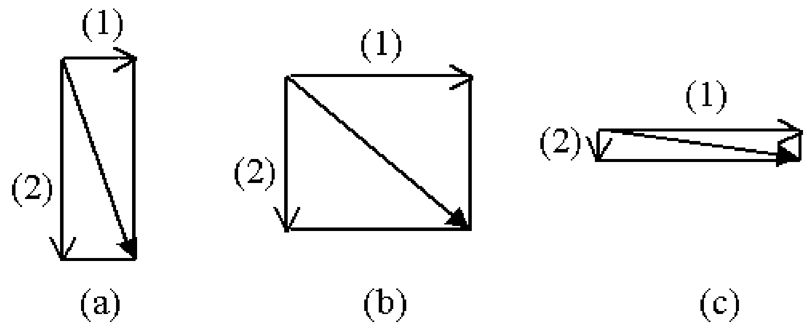


Figure 10 Relation between momentum vector and potential vector.

Vectors (1) and (2) represent momentum vector and potential vector, respectively. Cases (a), (b) and (c) represent gravity casting, squeeze casting and die casting (Elfakharany, 1999).

The calculation suggests that it is possible to visualize the fill pattern for slow processes using a single algorithm. The major difference is to determine the dominant flow or to calculate the contribution of different flows. Then the resistance and speed for each flow front voxel can be calculated and the fill pattern analysis can be performed.

The fill pattern produced by this algorithm was qualitatively compared against squeeze and gravity casting simulation and experimental results found in the literature. The patterns were very similar. A comparison for squeeze casting is shown in Figure 11 and Figure 12. At the beginning of cavity fill, the flow forms a bump at the gate. The flow front then becomes flat and steady.

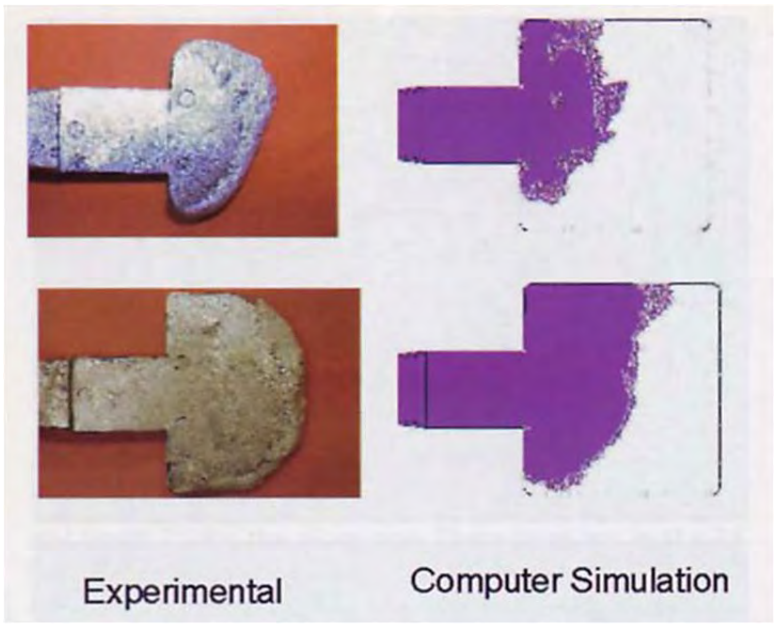


Figure 11 Experimental and simulation results (Procast) for 0.5 inch plate. [22]

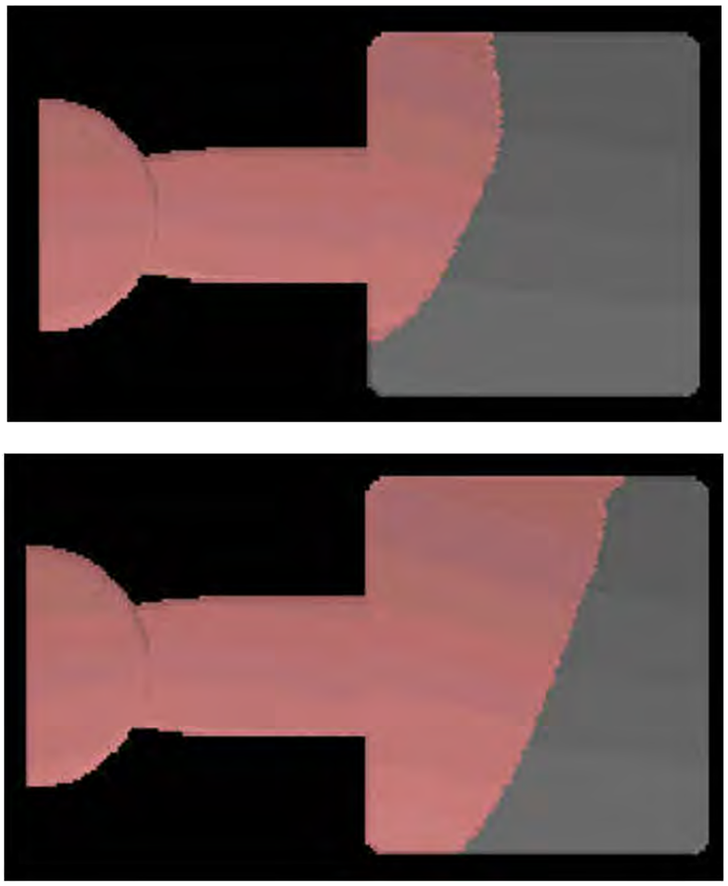


Figure 12 Visualization for 0.5 inch plate

3.2 Impact Spike Modeling

One goal of this research is to create a simple model that not only addresses the coupling effects between the plunger speed and metal pressure during the plunger deceleration, but also captures the cavity pressure efficiently without requiring the complexity of a 3-D simulation model. For this purpose, a multi-degree of freedom lumped-parameter model that integrates the injection system and components of die casting machine was developed and implemented in MatLAB's Simulink. The lumped-parameter model has many fewer degrees of freedom than a 3-D simulation model; therefore, insights can be achieved in a fast and inexpensive fashion but with a potential loss of detail. The other goal of this research is to evaluate the effects of plunger deceleration profiles on cavity pressure spike. With the aid of a shot monitoring system, the above lumped-parameter model can serve as a effective tool to accomplish this task.

3.2.1 Model Development

Figure 13 shows a schematic of a typical injection system that powers the piston that drives cavity filling. A simple 3-degree-of-freedom model (3-DOF) of this system expressed in terms of point masses, springs and dampers is shown in Figure 14.

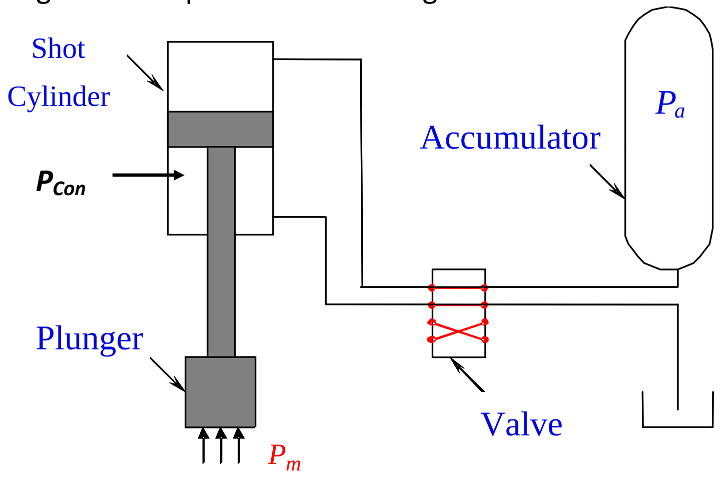


Figure 13 Schematic of an injection system

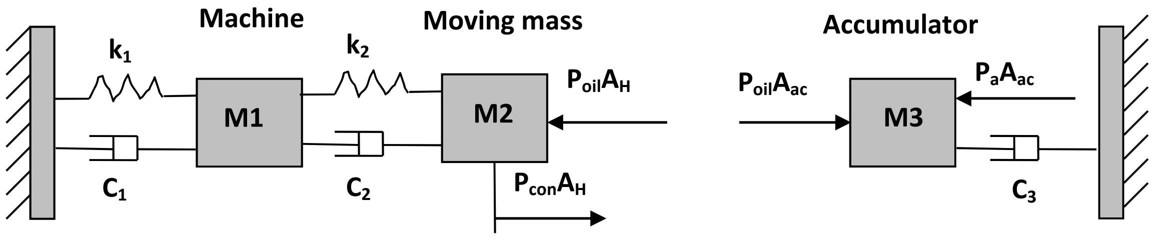


Figure 14 A 3-DOF Lumped-parameter model

The equations of motion associated with the model depend on the way the model will be used. If the model is to be used stand alone, it is necessary to include an approximation for the gate resistance to flow and the pressure increase after the cavity is filled must be explicitly represented. If the model is to be used in conjunction with numerical simulation, these approximations are not needed since the simulation will return the computed cavity pressure. The details of the model development are included in Appendix C.

The governing equations for the above system are

$$\begin{aligned} M_1\ddot{x}_1 &= -k_1x_1 - C_1\dot{x}_1 - k_2(x_1 - x_2) - C_2(\dot{x}_1 - \dot{x}_2) \\ M_2\ddot{x}_2 &= (P_{oil} - P_{con})A_H + k_2(x_1 - x_2) + C_2(\dot{x}_1 - \dot{x}_2) \\ M_3\ddot{x}_3 &= P_aA_{ac} - P_{oil}A_{ac} - C_3\dot{x}_3 \end{aligned} \tag{1}$$

where

M₁, *k₁*, and *C₁* – total mass, stiffness, and damping of the die and machine components

M_2 – total mass of moving components (liquid metal, hydraulic oil, and piston and plunger)

k_2 – equivalent stiffness due to the bulk modulus of liquid metal

C_2 – damping of injection system

M_3 – total mass of accumulator piston and hydraulic oil in the accumulator

C_3 – damping of accumulator system

P_{oil}, P_a and P_{con} – pressure of hydraulic oil, accumulator pressure, and controlled pressure against the hydraulic piston

A_H, A_{ac} – cross-sectional areas of hydraulic piston and accumulator piston

k_2 can be determined from the liquid metal bulk modulus E_B, the plunger tip area A_p, and cavity volume V_{cavity} as below

$$k_2 = \frac{E_B A_p^2}{V_{cavity}} \tag{2}$$

Any small change of the oil volume between the hydraulic piston and accumulator piston can cause a dramatic oil pressure change since the oil is essentially incompressible. Therefore, the dynamic coupling connection between M_2 and M_3 is captured by the following relationship:

$$\frac{dP_{oil}}{dt} = \frac{B_{oil}}{V_{oil}}(\dot{x}_2A_H - \dot{x}_3A_{ac}) \tag{3}$$

where B_{oil} is oil bulk modulus. Equations (1), (2) and (3) then can be used to solve the 3-DOF system with four unknowns $P_{oil}, \dot{x}_1, \dot{x}_2, \dot{x}_3$.

If the amount of hydraulic oil displaced by hydraulic piston and by accumulator piston is assumed to be the same during the injection (no air bubbles or other gaps form), then the displacements of the accumulator and hydraulic system pistons, x_2, x_3, are dependent due to volume conservation and meet the relationship

$$A_H x_2 = A_{ac} x_3$$

Defining $\beta = A_H / A_{ac}$, then the time dependent accumulator pressure can be evaluated by nominal accumulator pressure P_0, piston area ratio β and initial height of nitrogen in the accumulator, α :

$$P_a = \frac{P_0}{1+\alpha\beta x_2} \tag{4}$$

With this assumption the 3-DOF model can be simplified to a 2-DOF model, as shown in Figure 15,

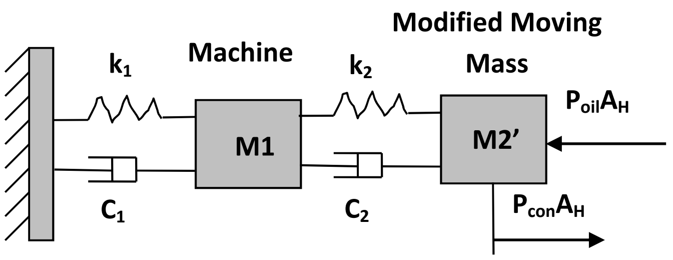


Figure 15 2 Degree of freedom lumped parameter model

and the equations of motion for the 2 degree of freedom model simplify to

$$\begin{aligned} M_1\ddot{x}_1 &= -k_1x_1 - c_1\dot{x}_1 - k_2(x_1-x_2) - c_2(\dot{x}_1-\dot{x}_2) \\ (M_2+\beta^2M_3)\ddot{x}_2 &= P_a\beta A_{ac} - c_3\beta^2\dot{x}_2 - P_{con}A_H + c_2(\dot{x}_1-\dot{x}_2) + k_2(x_1-x_2) \end{aligned} \tag{5}$$

To test the model, Matlab/Simulink was used to solve the above Ordinary Differential Equations (ODEs) that represent the injection process.

A good estimate of the system masses can be obtained from the mass of the plunger tip and rod, the shot piston and rod, the mass of die machine components, and the mass of the moving hydraulic fluid. The molten metal mass can be adequately estimated from the shot volume. Dry shot data were used to estimate the system and control parameters for velocity control. The damping was determined based on the time that the pressure oscillation of die and machine takes to come to rest. Then the control coefficients *K*'s were calculated according to the decay ratio and frequency from the dry shot plots.

The model uses programmed plunger speed as the input and calculates the piston slow and fast shot position and speed under the velocity control. Control switches to pressure control when the plunger reaches a specified position, for example, 99% full of cavity. A ramp-down pressure profile, which was determined by the pressure at switch point and the final static pressure, was applied and the system responses for the rest of plunger stroke and after-cavity-full were continually calculated.

The procedure is the same for coupling with numerical simulation, except that there is no need to use the Bernoulli equation and instead the simulation is used to compute the metal pressure directly.

3.2.1.1 Comparison of 1-, 2-, 3-DOF Lumped Parameter Models

Three lumped-parameter models with one, two, and three degrees of freedom respectively were implemented using MATLAB's Simulink. The tests and comparisons are based on these models. As expected, the 1 DOF model that lumped all masses into a single mass exhibited slightly stiffer behavior than the other two models. There was essentially no difference between the 2 and 3 DOF models as long as the accumulator damping is less than critically damped. For this reason, the 2 DOF is recommended for application.

The test results are shown in the following three figures.

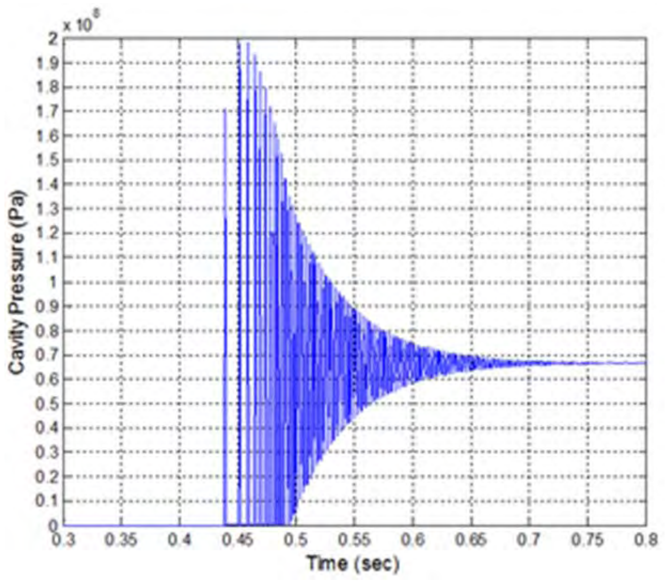


Figure 16 Cavity pressure response (1-DOF)

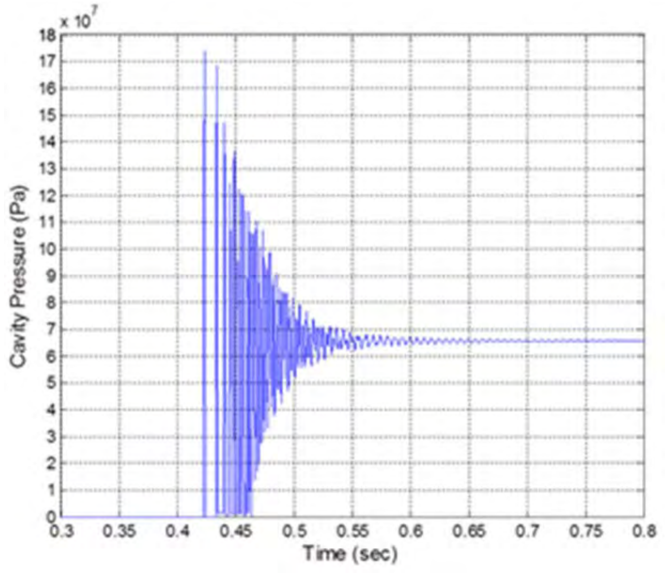


Figure 17 Cavity pressure response (3-DOF)

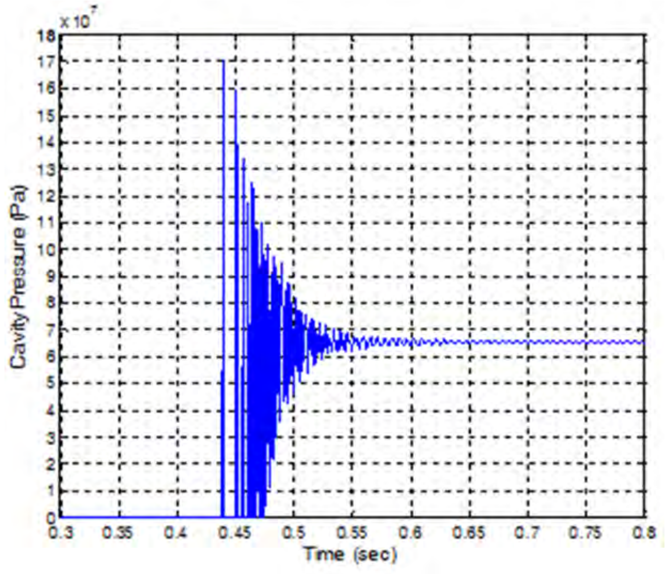


Figure 18 Cavity pressure response (2-DOF)

3.2.2 Deceleration Profiles

A decelerating plunger shot profile near the end of filling can decrease the impact pressure spike. The implementation of a deceleration profile is practically very difficult to accomplish due in part to the response times of the hydraulic system involved, but it is useful to explore the question of deceleration to help quantify the benefits that could be obtained with a change in standard operating procedure. The effect of deceleration on the cavity pressure spike was explored through a simple set of computational experiments.

Four process parameters that can control the plunger deceleration were investigated: deceleration start time, plunger zero velocity position, plunger position when velocity control switches to pressure control, and rise time to full accumulator pressure. The descriptions of the factors along with their levels are shown in Table 1. A graphical depiction is shown in Figure 19.

A 27 run Box-Behnken response surface experimental design was chosen based on the four factors [12]. The optimal parameters were found to be a linear deceleration profile with all factors at the low level. The cavity pressure responses when using optimal levels is plotted in Figure 20. With linear deceleration profile, the maximum pressure peak is about 1.23e8 Pa. Without plunger pre-deceleration and ramp-up hydraulic pressure, the impact pressure would be as high as 2.60e8 Pa. Therefore, as expected, deceleration significantly reduces the spike suggesting again that just about any deceleration will help to reduce the spike.

Table 1 Design of Experiments

Factor	Description	Low Level	Medium Level	High Level
A	Deceleration start time	0.41644 s	0.41788 s	0.41932 s
B	Plunger zero velocity position	26.432 cm	26.449 cm	26.462 cm
C	Plunger position where velocity control switches to pressure control	26.371 cm	26.401 cm	26.431 cm
D	Rising time of hydraulic pressure from zero to full accumulator pressure	0.01 s	0.03 s	0.05 s

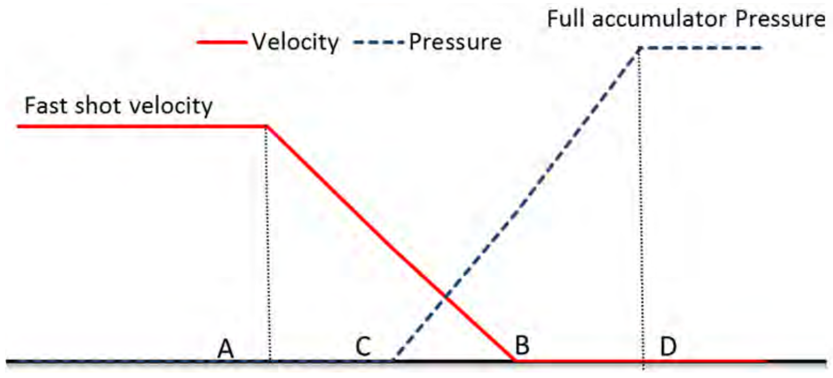


Figure 19 Illustration of Factor Definitions

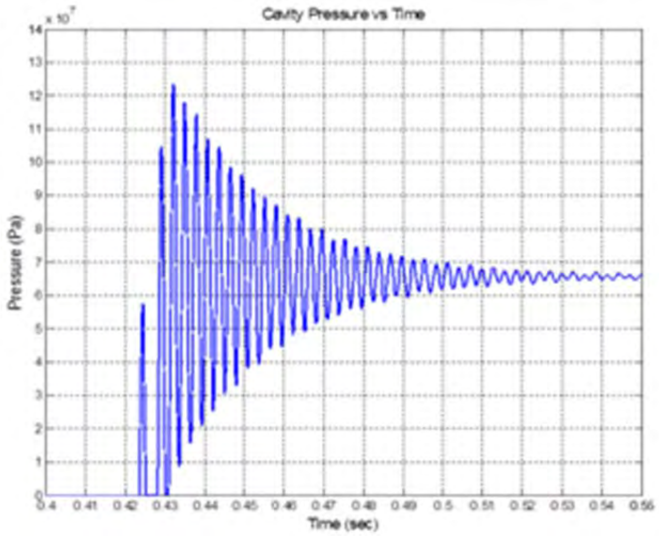


Figure 20 Cavity Pressure with best linear deceleration

3.2.3 Issues with Coupled Simulation

One of the objectives of the work was to develop a model that could be coupled with numerical fill simulation to provide a more accurate prediction of cavity pressure near the end of fill. Toward this end a test simulation was developed and run using Flow 3D. Unfortunately, computational issues related to the volume of metal make this approach impractical.

A fluid flow model to simulate injection was developed. The system is shown in Figure 21. A typical simulation tracking fluid velocity and pressure was run. Previous experience, and discussions with the software vendor have confirmed, that there can be issues with “volume loss.” Tracking the flow front is a difficult task and as the simulation evolves the total volume of fluid changes due to computational errors. This loss creates serious problems for pressure calculations at the end of fill since volume-distance-speed relationships are not preserved. Additional tests with different meshing strategies and shot velocity have disclosed that the problem exists over a wide range of modeling and simulation parameters. Volume loss data for several test simulations are shown in Figure 22.

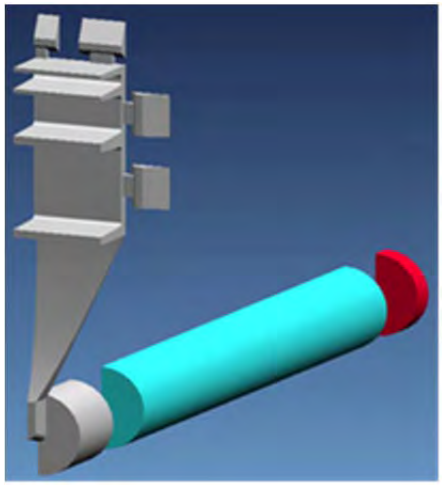


Figure 21 Test Simulation

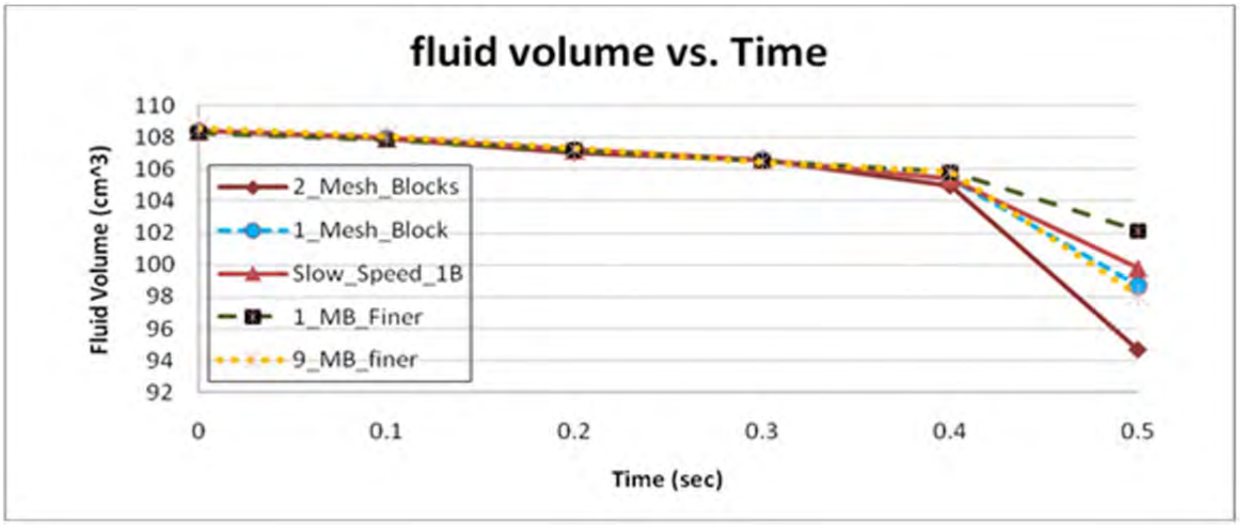


Figure 22 Fluid Volume Changes

To save simulation run time, a simple geometry was used to further investigate the fluid volume issue, as shown in Figure 23. Several cases, including fast speed/slow speed, fine mesh/course mesh, and Turbulence/Laminar model, were simulated and the fluid volume changes during simulation were recorded (Figure 24). All models and conditions again showed volume loss.

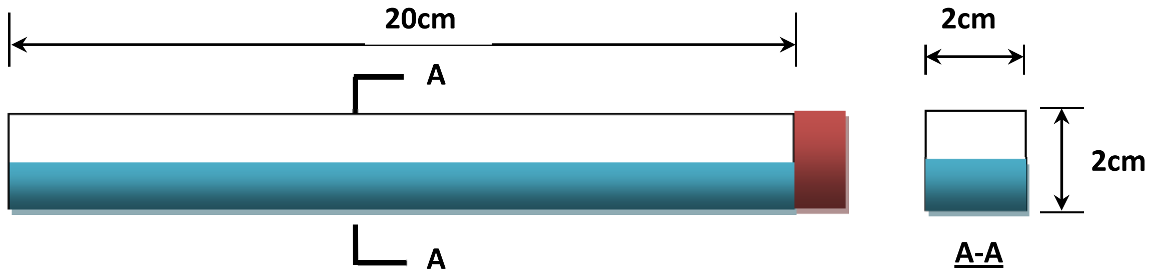


Figure 23 Test Block Geometry

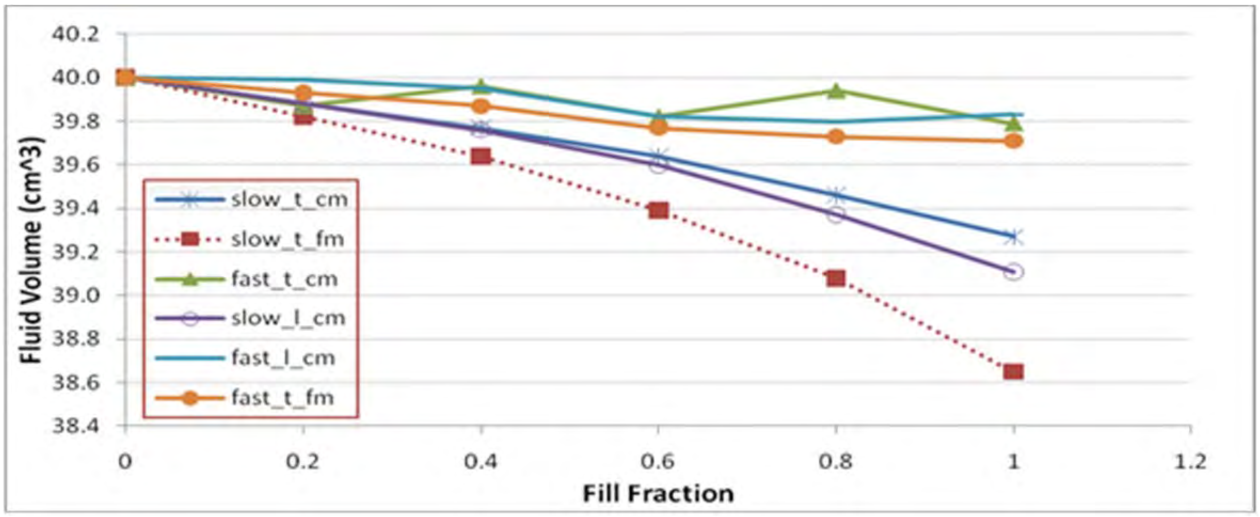


Figure 24 Fluid Volume Changes

The results show that no simulation approach is immune from the problem, and further, the ideal mesh for the application may not be the best in terms of volume loss.

In summary, the exploration of the volume loss issue suggests that it is not practical at this time to utilize fill simulation to establish the pressure distribution within the cavity. Therefore, the coupling of the lumped model with simulation will not provide reasonable pressure inputs for a structural die deflection, parting plane separation, simulation. Consequently, the best approach available to account for the dynamic pressure at the end of fill is to use the lumped model directly without incorporation of a fill simulation.

3.3 Thermal and Cycle Optimization

This task builds on recently developed techniques that efficiently compute an approximate equilibrium die temperature distribution without the need to simulate 50 or more casting cycles in order to ensure that the die temperature distribution is in quasi equilibrium. A summary of the equilibrium temperature concept and its use for design is provided in Appendix E.

The objective was to develop techniques to optimize heat balance in the die by mathematically adjusting heating or cooling is needed and iteratively improving the actual locations of heating or cooling lines. The implementation of mathematical techniques to automatically accomplish this objective was intractable given the evolution of the project. However, the approach was demonstrated using manual iterations and a description is reported here.

Work on die modification/configuration prerequisite to mathematical optimization was also completed and a detailed summary is included in Appendix F.

3.3.1 Cooling and Cycle Optimization

This study was performed in conjunction with an industrial partner who experienced difficulties during production with a small hot chamber machine and die. The problem served as an ideal case study to demonstrate design optimization.

The machine was a 66 NT Techmire hot chamber die casting machine. Four parts per shot are produced using a multi-cavity die design. The entire four cavity die was about 15.24 cm (6 in) wide by 15.24 cm (6 in) tall. The casting had a total mass of 42 g and made of ZAMAK 5. The casting cycle with the original design was 10.2 sec. The two sides of the die are shown in Figure 25.

During operation, machine tie bolts failed in seemingly random fashion. Tie bolts have a similar function as the tie bars have in a cold chamber machine holding the machine structure together and are stretched when the clamping force is applied. Because of the cyclic nature of the process the tie bolts experience cyclic loads during production. Thermal and stress analysis showed that the parting surface of the die swelled outward due to heat growth causing a high and unbalanced loading on the tie bolts, leading to early failure.

A thick section analysis of the casting was performed using CastView and is shown in Figure 26. It is very clear from the image that the orientation of the two parts closest to the gate (left side) is such that the heaviest section is close to runner system. The combination of the heavy section and runner which contains a significant amount of hot material places a heavy heat load near the center of the die. This is consistent with the heat growth pattern that was experienced.

The layouts of the cooling systems for the cover and ejector dies are shown in Figure 27 and Figure 28 respectively. It is clear from the combination of the thick section analysis and the cooling layout that there is excess cooling on the right side of the ejector die where there is little heat, and little cooling near the major hot spots. Also, there is better cooling in the cover die, but the need is in the ejector side. A map of the ejector die temperature is shown in Figure 29 and the hot spot closely matches the regions identified above.

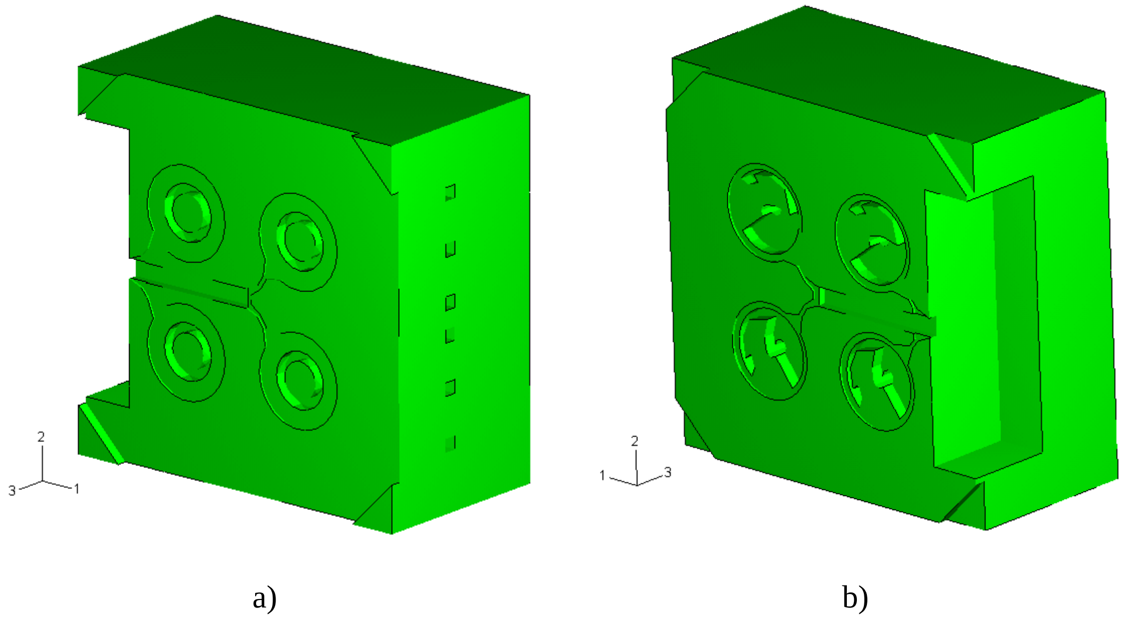


Figure 25 Original dies design, a) Cover side, b) Ejector side

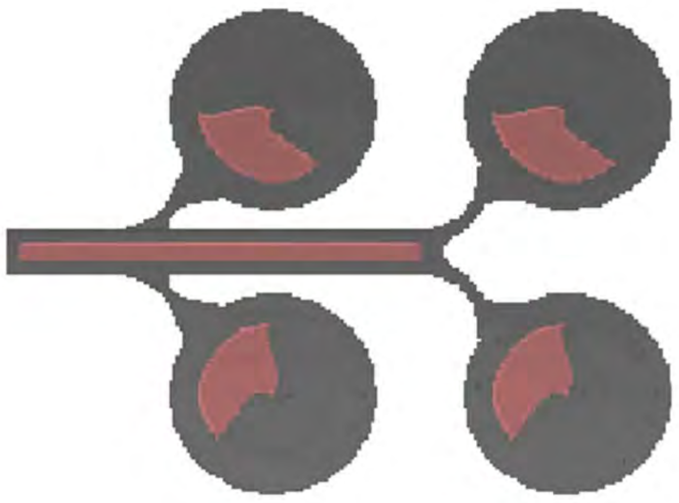


Figure 26 Thick Section Analysis, Original Design

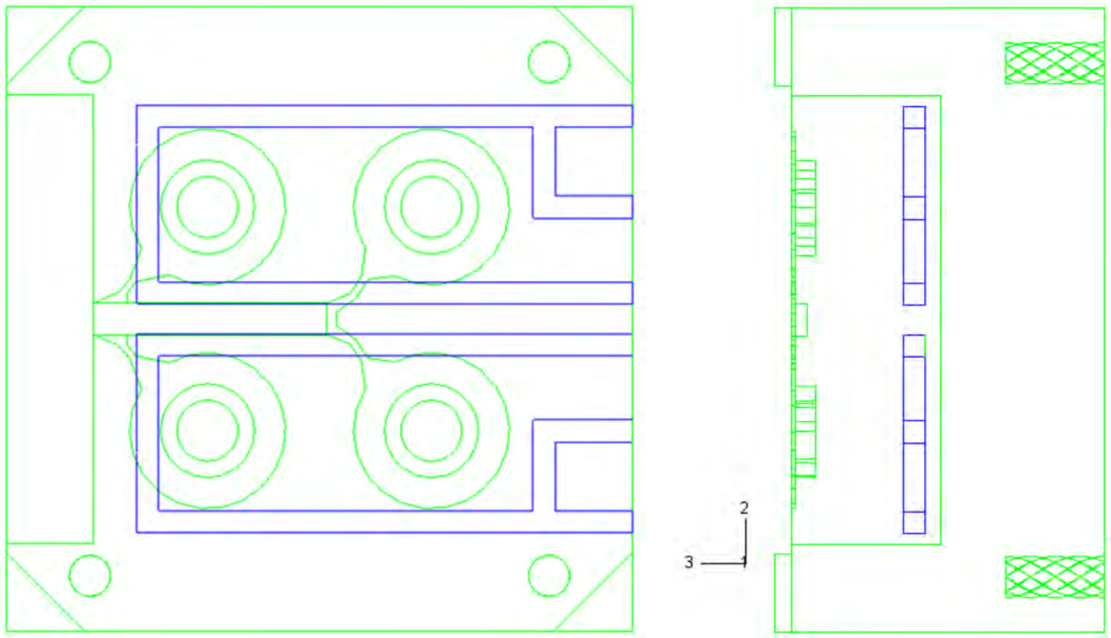


Figure 27 Cover die cooling layout

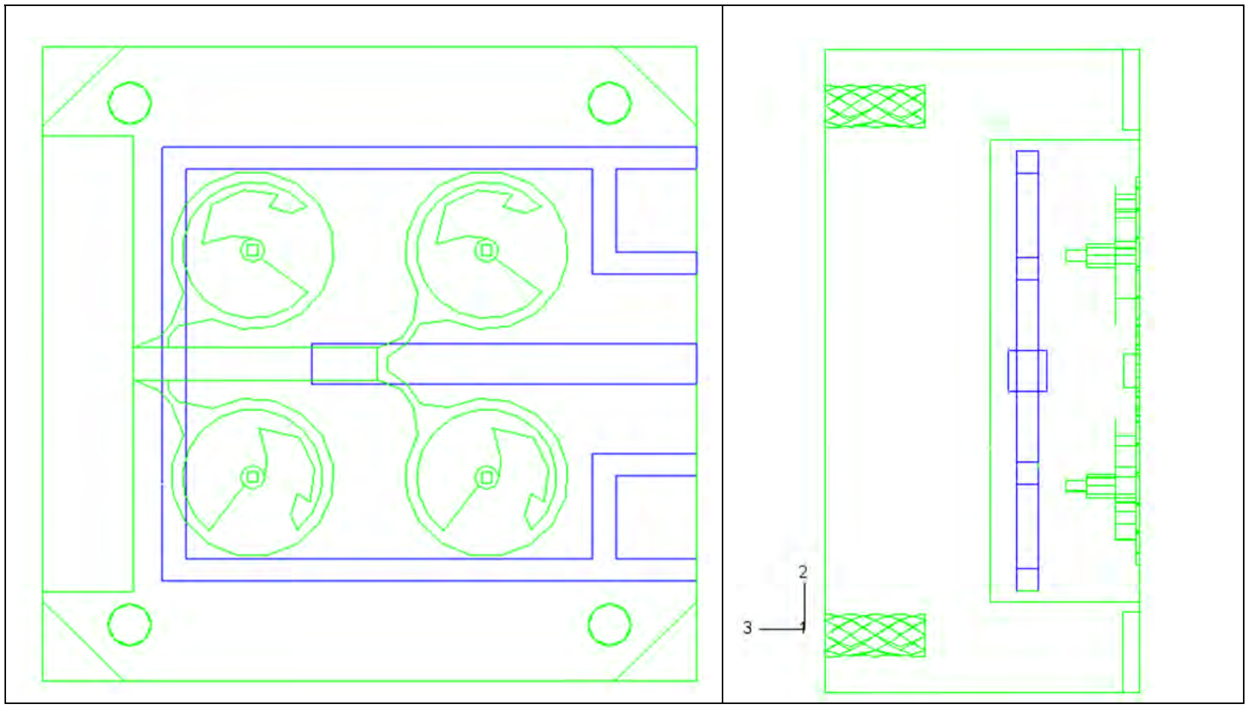


Figure 28 Original ejector die cooling layout

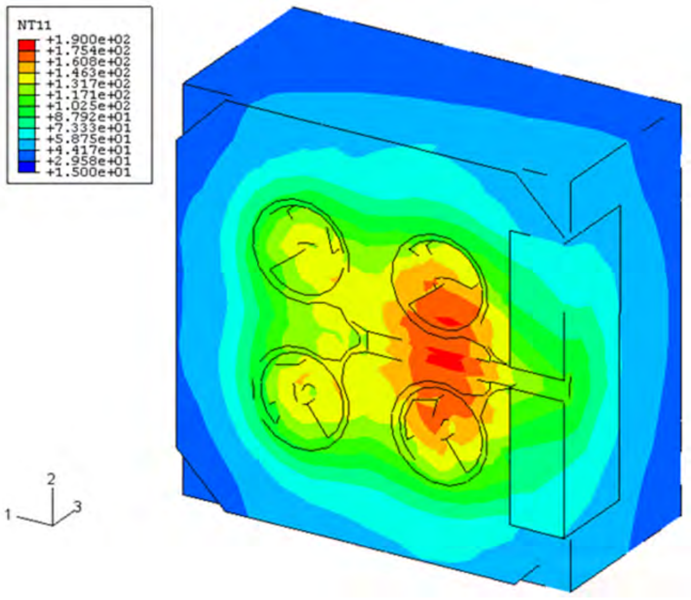


Figure 29 Ejector die equilibrium temperature

Clearly, better cooling is possible by placing the lines near the source of the heat, and the heat load can be better balanced by rotating the heavy sections near the inlet away from the runner as shown in Figure 30. This configuration maintains a consistent gate location for all cavities and redistributes the heat a little better. The final cooling line layouts arrived at after several iterations are shown in Figure 31 and Figure 32. This design brings a loop of cooling directly behind each cavity.

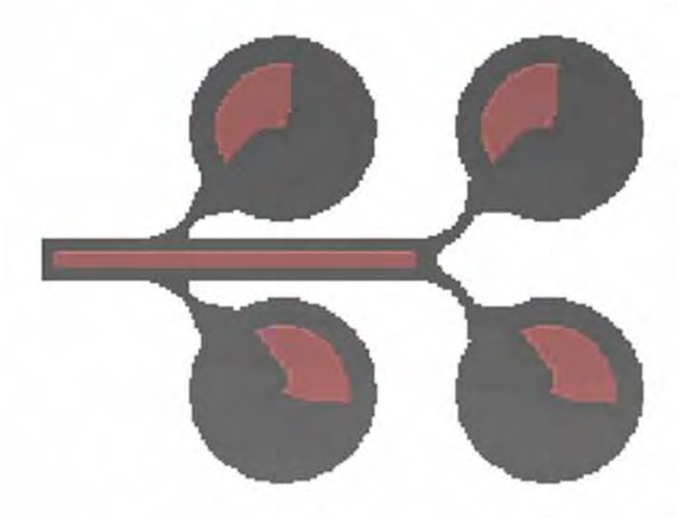


Figure 30 Heavy Section for Modified Layout

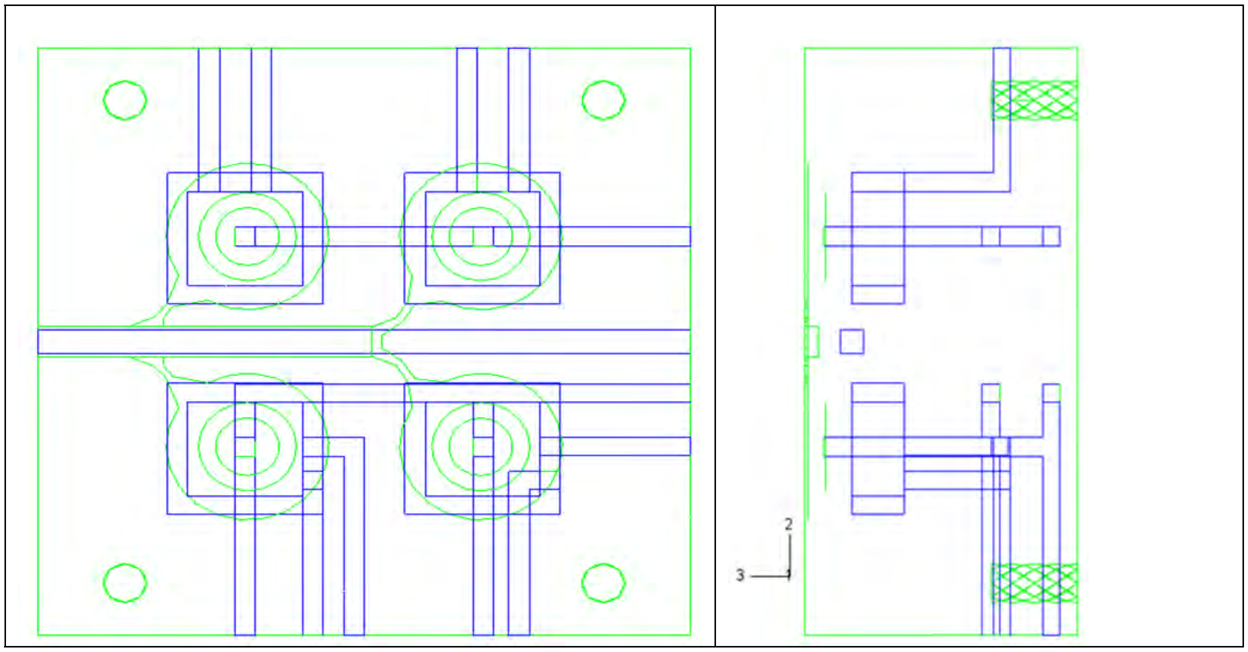


Figure 31 Final cover side cooling layout

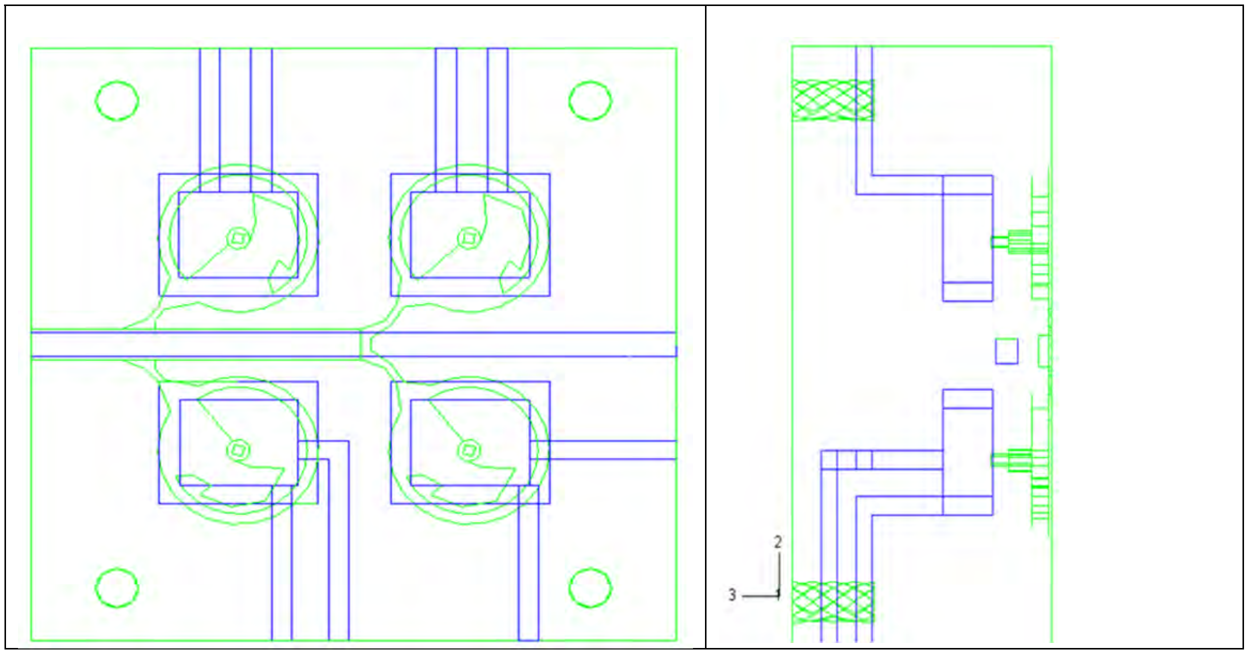


Figure 32 Final ejector side cooling layout

The temperature distribution on the ejector side is shown in Figure 33. The hot spots are now more consistent across cavities, and particularly important for parting plane growth, the hot regions are in interior of the cavity and not on the parting surface. This may affect the part dimensions slightly, but the nearly uniform temperatures on the parting surfaces means that tie bar loading will be much more balanced.

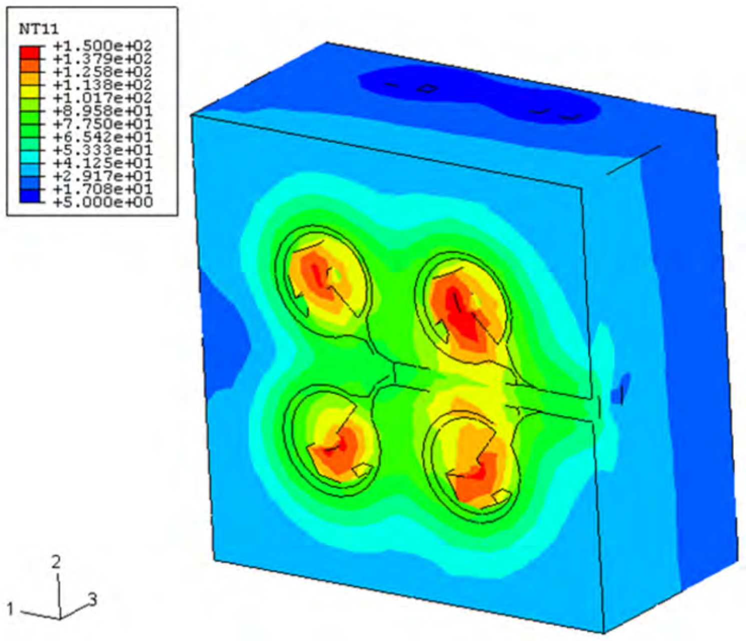


Figure 33 Ejector equilibrium temperature distribution

The die layout, coupled with a reduction in cooling water temperature, has a significant effect on the die performance. The cycle time with the modifications was reduced to 6.3 seconds, a 38% improvement, and there is no non-uniform tie bar loading to cause production interruptions. Such improvements may not be possible in every case, but this study clearly demonstrates that a few iterations with the cooling system and the orientation of the part in the die can have a dramatic impact on productivity due both to cycle time improvement and process continuity. The later, in particular, means less die warm up recovering from failure and repair. Each stop and restart means wasted energy and this energy is saved with an optimized die.

3.3.2 Automated Geometry Modeling

As illustrated in the case study described in the previous section, optimizing and analyzing a die requires the ability to manipulate the part geometry relative to the die geometry. The required operations include parting surface placement, undercut detection and cavity geometry construction. The CastView program uses STL files as the basic geometry input medium primarily to ensure CAD system independence. The lack of topological information in STL data means that these neighbor relationships must be reconstructed. Once the topology is available, the data can be used, along with user provided information about the desired die opening direction and parting surface location, to define the die cavity surfaces. The master's work of Li Shen developed algorithms for many of these tasks. Those interested in the details of this work will find a detailed description of the problems and algorithms in Appendix G.

4. Benefits Assessment

This new technology was predicted to result in an average energy savings of 1.83 trillion BTU's/year over a 10 year period. Current (2011) annual energy saving estimates over a ten year period, based on commercial introduction in 2012, a market penetration of 30% by 2015 is 1.89 trillion BTU's/year by 2022.

Along with these energy savings, reduction of scrap and improvement in yield will result in a reduction of the environmental emissions associated with the melting and pouring of the metal which will be saved as a result of this technology. The average annual estimate of CO_2 reduction per year through 2022 is 0.037 Million Metric Tons of Carbon Equivalent (MM TCE).

Estimates of energy savings are based on the following assumptions:

- There was an estimated 1.6 million tons of die casting shipments in 2000 (U.S. Department of Commerce, U.S. Census Bureau, Current Industrial Reports, Iron and Steel Castings, MA331E (00)-1.)

- The die casting industry will grow at a rate of2.5% per year (AFS Metalcasting Forecast & Trends 2002 (Oct. 2001))

- Baseline energy consumption for die casting is 23.4 million Btu per ton = 6858kWh

- The results of this project will provide the ability to design better dies which will in turn reduce scrap and improve operating efficiency in contributing to a combined 8% improvement in energy efficiency.

Assumes 3412 Btu/kWh (Source: DOE/EIA, Monthly Energy Review)

Approximately 25% of scrap and 75% new material are used during melting. The average price for Aluminum scrap in 2001 was .49 a pound or $980 a ton (Source: American Metal Market). For new aluminum it cost about $0.70 a pound or $1400 per ton. (Source: U.S. Geological Survey, Mineral Commodity Summaries, Jan 2002) A majority of die castings are aluminum therefore aluminum price data was used for these calculations

5. Commercialization

This work will be incorporated into NADCA's Productivity Improvement Suite. Specific commercialization steps include:

- The impact spike models are in the process of being implemented as a web-based application that can be accessed from the NADCA design tools web site. A user will be able to perform a variety of 'what ifs' to better understand the relationship between machine, process parameters, and dynamic pressure.
- The impact spike model will be included as a module in the next release of the NADCA PQ^2 program. This will be part of the tools supporting optimal machine selection based on part requirements.
- The fill visualization improvements are part of version 3 of the CastView program that will be released by NADCA early in 2012. The slow fill component will be included but gating will have to be imported with the part.
- The thermal calculations that support design optimization of the sort described in section 3.2.1 will be included in version 4 of CastView that will occur during the third quarter of 2012.
- This work has been reviewed twice a year in the NADCA Computer Modeling Task Group and once per year in the NADCA Research and Development Committee throughout the duration of the project. Both groups are composed of members of the industry and these vehicles provide a mechanism to disseminate information as it develops as well as provide guidance to the research.

6. Accomplishments

This work has developed simple models and extended existing models that support design decision making by part designers, die casters and die builders. The most important feature of all of this work is that the computational burden is very low encouraging the exploration of more design options within the available time. In addition, the output is almost always graphical maximizing communication among the interested parties.

The following papers based on this project were published and presented at casting industry technical congresses.

D. Wang, R. A. Miller, "Improvement of Geometric Reasoning of Fill Pattern Analysis," *108th Metal Casting Congress*, Paper T01-053, North American Die Casting Association, Rosemont, Illinois, June 2004.

H. Xue, K. Kabiri-Bamoradian, R. A. Miller, "Modeling Dynamic Cavity Pressure and Impact Spike in Die Casting,", Paper T05-033, *NADCA Transactions 2005*,North American Die Casting Association, Wheeling, Illinois, April 2005.

R. A. Miller, D. Wang "Design Support for Tooling Optimization," Paper T05-102, *NADCA Transactions 2005,*North American Die Casting Association, Wheeling, Illinois, April 2005.

H. Xue, K. Kabiri-Bamoradian, R. A. Miller, Y. Wang "Modeling Dynamic Pressure in High Pressure Die Casting Using Lumped Parameter Models,", *Transactions of 2011 Die Casting Congress and Tabletop*, North American Die Casting Association, Wheeling, Illinois, 2011.

The fourth of these papers, the Xue, Bamoradian, Miller, Yang paper, received the 2011 Congress Best Paper Award.

The research partially supported the work of the following students:

Dongtao Wang, PhD dissertation

Hongyu Xue, PhD dissertation

Li Shen, MS thesis

7. Conclusions

The project has successfully extended and demonstrated modeling capabilities that support design decision making for high pressure die casting. It has also developed fill visualization tools suitable for slow fill processes that enable very fast assessment of the fill pattern associated with these processes.

Several improvements in the qualitative reasoning techniques used for fill visualizations have been made to incorporate (qualitatively) fill speed and flow resistance. This makes the results more accurate for castings with ribs and similar structures. Simple techniques to incorporate the relative contribution of viscosity, inertia and gravity were developed and this enables use of the fill visualization techniques for more processes.

Multi-degrees of freedom lumped-parameter models that incorporate the injection system and die casting machine components were established. This modeling provides a frame work to develop strategies to evaluate dynamic effects without requiring the complexity of a full dynamic finite element model. Die design, machine design, accumulator design, and shot profile all play a key roles in reducing the cavity pressure spike developed at the end of injection. The lumped-parameter model provides a quick and inexpensive tool to estimate/evaluate the different design parameters. Given that numerical problems tend to amplify near the end of fill with numerical fill simulations, it is currently not practical to couple the lumped model with a fill simulation to calculate metal pressure. At present, the lumped parameter model is probably the best option available to estimate the dynamic evolution of cavity pressure.

The use of the equilibrium temperature distribution procedures as a tool for the iterative improvement of die thermal balance and cycle time was demonstrated. A case study showed that optimizes the part orientation in the die and matching cooling line locations to source of heat significantly improves die performance and lowers the feasible cycle time. These are not new observations, simulation has provided this capability for years, but the possibility of using of the very efficient, low computational burden, techniques that were presented is new and offers an important benefit to the die casting industry.

8. Recommendations

Much of the work reported here built upon the existing infrastructure provided by the CastView program. This program converts STL data to voxels as the basic representation for analysis and rendering of results. In recent graphics hardware has improved greatly and now many of the required operations can be performed in hardware. The hardware implementation is sufficiently fast to allow real time rendering, something the current system cannot do. The utility of the program would be greatly improved if the analysis and display functions were updated to take advantage of current hardware capabilities.

9. References

1. Li, Y. B. and Zhou, W., Numerical Simulation of Filling Process in Die Casting, Materials Technology, Vol. 18, No. 1, 2003, pp. 36-41.
2. Jia, Liangrong; Xiong, Shoumei and Liu, Baicheng Study on numerical simulation of mold filling and heat transfer in die casting process Journal of Materials Science and Technology, Vol.16, No. 3, 2000, pp. 269-272.
3. McLaughlin, Martin; Kin, Chung-Whee and Backer, Gerald, The Importance of Trapped Gas Analysis in Magnesium Filling Simulations, Die Casting Engineer. Vol. 47, No. 3, 2003, pp. 32-36.
4. Heine, R. W. and Uicker, J. J. Risering by Computer Assisted Geometric Modeling, Transactions of the American Foundrymen's Society, Vol. 91, 1983, pp.127-136.
5. Ma, Y., Qualitative Geometric Reasoning for Thermal Design Evaluation of Die Casting Dies, Ph.D. Dissertation, 2000, The Ohio State University.
6. Elfakharany, Essameldin F., Qualitative reasoning for filling pattern in high-pressure die-casting and gravity-driven casting, PhD dissertation, 1999, The Ohio State University.
7. Miller, R. Allen, Lu, Shao-Chiung and Alexander B. Rebello, Simple visualization techniques for die casting part and die design, Final report, DOE project # DE-FC07-95ID13362, 1998.
8. Rebello, A. B., Visualization of the filling of die-casting dies, PhD dissertation, The Ohio State University, 1997.
9. Ahuett-Garza, H., Miller, R.A., “Die casting die deflections: Computer simulation of causes and effects”, Transactions of the 19th International Die Casting Congress & Exposition (1997)
10. [Branden, J., Olmstead, P. and Kuhn, C.W., “Plunger Deceleration of a Die Cast Shot End – Applying Peter Olmstead SoftSHOT^{TM} Simulation Software”, NADCA Congress (2002)
11. Choudhury, A.K., “Study of the Effect of Die Casting Machine upon Die Deflections”, The Ohio State University, Master Thesis (1997)
12. Montgomery, D.C., Myers, R.H. “Response Surface Methodology: Process and Product Optimization Using Designed Experiments” , ISBN 0-471-41255-4 (2002)
13. Kesavan, V., “Development of Computer Simulation Models for Die Casting Studies”, The Ohio State University, Master Thesis (1999)
14. Mickowski, J.R. and Teufert, C.E., “The Control of Impact Pressure in the High Pressure Die Casting Process”, Transactions of the 17th NADCA Congress & Exposition (1993)
15. Ragab, A., “Sensitivity Analysis of Casting Distortion and Residual Stress Prediction through Simulation Modeling and Experimental Verification”, The Ohio State University, PhD dissertation (2003)

16. Savage, G., Gershenzon, M. and Rogers, K.J., “The Role of Pressure in High Pressure Die Casting”, NADCA Congress (2001)
17. Venkatasamy, V., Brevick, J., Mobley, C. and Pribyl, G., “Die Cavity Sensors for Monitoring Die Casting Processes”, Transactions of the 19th International Die Casting Congress & Exposition (1997)
18. Tong, K.K.S., Hu, B.H., Niu, X.P. and Pinwill, I., “Cavity Pressure Measurements and Process Monitoring for a magnesium Die Casting of a Thin-Wall Hand-Phone Component to Improve Quality”, Journal of Materials Processing Technology, Vol. 127, pp. 238-241 (2002)
19. Allchin, T., Quality assurance with modern CNC die casting system, *Die Casting Engineer*, 1990, March/April, p34-38.
20. Xue, H., Kabiri-Bamoradian, K., Miller, R. “Modeling Dynamic Cavity Pressure and Impact Spike in Die Casting”, CastExpo/NADCA (2005)
21. Reikher, A., Gerber, H., and Starobin, A., “Multi-Stage Plunger Deceleration System”, CastExpo/NADCA (2008)
22. Wallace, John F.; Chang, Qingming and Schwam, David, Process Control in Squeeze Casting, *Die Casting Engineer*, 2000, November/December, p42-48.

10. Appendices A - F

Appendix A:	Improvements on Geometric Reasoning of Fill Pattern Analysis for Die Casting
Appendix B:	Visualization of Slow Fill Processes
Appendix C:	Modeling Dynamic Cavity Pressure and Impact Spike in Die Casting
Appendix D:	Modeling Dynamic Pressure in High Pressure Die Casting Using Lumped Parameter Models
Appendix E:	Design Support for Tooling Optimization
Appendix F:	Automated Geometric Modeling of Die Components

Appendix A: Improvements on Geometric Reasoning of Fill Pattern Analysis for Die Casting

Improvement on Geometric Reasoning of Fill Pattern Analysis for Die Casting

Dongtao Wang and R. Allen Miller
The Ohio State University, Columbus, Ohio

Copyright 2004 American Foundry Society

ABSTRACT

Fill pattern is always a key issue in manufacturability analysis for die casting engineers. The Ohio State University previously developed a quick algorithm to visualize the cavity fill pattern using geometric reasoning without solving fluid flow and conservation equations. The reasoning method was used in the CastView program. Though the algorithm has proved efficient, there is room for improvement and a new algorithm has been designed. The calculation of a new direction vector when flow hits an obstruction was redesigned by considering the original vector, available empty voxels and more new change of direction angles. Multiple gate locations can be specified so that cavity fill can be started simultaneously from multiple gates. A flow resistance factor was defined that affects flow speed and is determined by cavity geometry and the local flow vector. The direction and speed of flow front thus determine the fill pattern. Some efforts were also placed on efficiency improvement. The comparison with our results and commercial software results based on numerical simulation suggests the adequacy of the improvements.

INTRODUCTION

It is well known that the filling process in die casting is one of the most important factors affecting the product quality. The metal flow, if not controlled properly, can cause flow-related defects. The most common flow-related defect may be gas porosity, which usually happens at the last region to be filled in cavity. Thus it is very desirable for die casting engineers to know the fill pattern prior to the production.

At present, the prevailing method to estimate a fill pattern is numerical simulation. A simulation typically sets up equations based on mass conservation, momentum conservation and energy conservation then solves the equation system [1~3]. Due to the large number of equations, equation solving is usually a time consuming task. Furthermore, the simulation systems usually require the user to have considerable experience and good understanding of fluid dynamics. These reasons limit the number of "what-ifs" that can be performed and also limit numerical simulation's utility in the early process design stage.

One of the alternative methods is qualitative reasoning for fill pattern prediction. This method is based on the geometric characteristics of cavity so it only requires the input of part and gate geometry and does not require fluid dynamics background of the user. Moreover, there are no complex equations to be solved so the analysis can be finished quickly, which is very desirable for the early stage design. The pioneers using geometric reasoning for die casting process analysis are Heine and Uicker [4], Ravi, Kotschi and Upadhya [5]. The CastView research group at The Ohio State University developed a qualitative reasoning algorithm for fill pattern prediction [6]. The input is part geometry in STL or stereo lithography format (a triangular faceted representation) on which the user can define the gates. The STL part is first discretized into a voxel model (a cubic box representation) and the fill pattern is determined based on computational geometric reasoning with typical run time of a few minutes.

The application of fill pattern analysis in CastView has proved that this is a simple and efficient way to predict fill pattern in die casting cavity. Though the accuracy level is not as high as that of numerical simulation and the information provided is incomplete, it provides a quick but reasonable solution for the fill pattern analysis. There were, however, occasional problems of the analysis in some cases. Therefore, it is very necessary to improve the original algorithm to obtain a more accurate result. This paper presents the method of improvement and compares the new results with old results and with those of numerical simulation.

ORIGINAL ALGORITHM

This section briefly introduces the original algorithm implemented by CastView. In the analysis, the cavity geometry is first discretized into a voxel model, where a voxel is a small cubic box and a set of voxels represent the part geometry. This is similar to geometry discretization in the finite difference method.

In the fill visualization algorithm, some assumption and simplifications were made as follows:

- The flow does not change direction until it is obstructed by the cavity wall or a previously filled region.
- When the flow encounters an obstruction, it will find available empty voxels within the local neighborhood.
- Conservation of volume is considered within neighborhood voxels.
- An empty voxel can be filled by metal exiting one or more voxels, i.e., the new voxel can have one or more parent voxels.
- Due to the discrete voxel space, there are limited directions for flow from a voxel. An interior voxel can have five sets of flow vectors but a gate voxel has fewer vectors.
- If the flow front is completely obstructed, the flow direction is reversed and each voxel acts as a point source of a wave.

In discrete voxel space, an input flow vector is "rounded" to the nearest discrete vector available. For example, in Fig. 1, the output vector of voxel a must be rounded to available $\overrightarrow{ab}$ or $\overrightarrow{ac}$, which are based on one layer outer voxels (first set). If the second outer layer is counted, the vectors $\overrightarrow{ad}$ or $\overrightarrow{ae}$ are also available (second set). To keep both vector accuracy and a small computation domain, at most there are five sets of vector available (vectors pointing to the fifth outer layer).

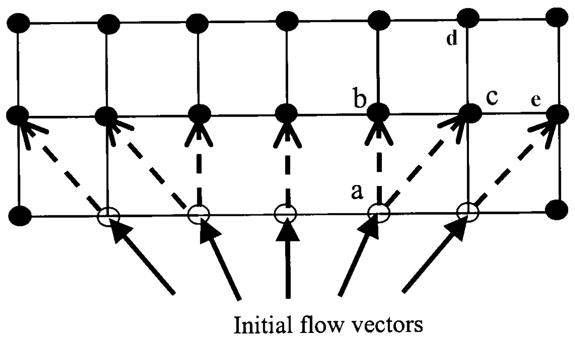


Fig. 1 Possible output vector from voxel a [6]

As the flow advances, it may hit obstructions, which can be cavity wall or pre-filled voxels. When the flow is obstructed, the algorithm does the following:

- Search the first set of nearest voxels for empty voxels;
- If there are empty voxels, distribute the voxel material over the empty voxels and calculate new flow vectors;
- If there are no empty voxels in the first set, search the second set;
- If there are no empty voxels in all five sets, consider the path to be completely blocked.

Problems with the original algorithm may include:

- Speed not considered. Flow has different directions but has the same speed.
- Flow resistance not considered. Flow resistance is defined as effect of local geometric characteristics on the flow. For example, flow at a thin section generally has larger resistance than at thick section.
- Vector calculation at an obstruction is incomplete. In the original algorithm, when a flow hits the obstruction, it would disperse to all nearby empty voxels.
- Multiple gates not considered. The flow may start cavity filling at one gate earlier than others. These existing problems may cause errors and inconsistencies and they should be improved with the new algorithm.

NEW ALGORITHM

The basic idea is to correct the problems mentioned earlier but not to increase computation time or memory requirements significantly since the efficiency is a principal goal. The following improvements have been accomplished:

1. Speed calculation

In the fill pattern algorithm, a list is used to store the flow front voxels. The cavity voxels can be classified into three groups, filled voxels, empty voxels and flow front voxels. The initial flow front voxels are gate voxels with the initial incoming vector. In each calculation step, the flow front voxels will fill some empty voxels and generate new flow front voxels. This calculation is continued until the whole cavity is filled.

A flow front voxel had direction (vector) but no speed in the original algorithm. In another word, all flow front voxels had the same speed. In the new algorithm, speed is included (qualitatively) and each flow front voxel may have a different speed. Both speed and the direction of flow may change during cavity filling. There are three possibilities that cause speed change:

a) When the flow hits an obstruction. As the flow vector changes, the speed drops due to energy loss;
b) When a voxel is filled by more than one voxel. The speed of outgoing voxel is calculated based on the speed of incoming voxels;
c) When flow resistance (discussed in detail later) changes due to local geometry the flow speed then may change.

The speed is also rounded to a small number of speed levels to save the computation and storage expense. This is reasonable since we are calculating the fill sequence and not the dynamic change with time. Furthermore, we can freely scale all the speeds of flow front voxels at the same time, which allows us to normalize the speeds to a fixed range. For example, if all the speeds are too small or too large, we can always normalize them to the range of 0 ~ 100.

2. Flow resistance

A flow resistance for each flow front voxel is defined based on local geometric characteristics. One example of flow resistance is shown in Fig. 2. Flow is coming from A and will fill B earlier than fill C since flow resistance at C is larger than that at B. Flow speed at C is reduced due to larger flow resistance. Apparently the flow resistance is related to cavity openness. The more open, the smaller the resistance. In addition, it is also related to the flow vector. For example, in Fig. 3, flow B will have larger resistance than flow C when they are filling the two ribs because there is an angle between flow B and the rib, which makes it hard for flow B to fill the rib.

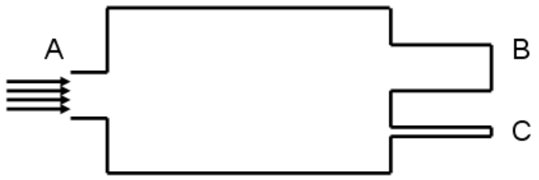


Fig. 2 Flow will fill B earlier than fill C

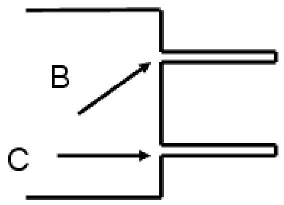


Fig. 3 Flow B has larger resistance than C

An initial definition of cavity openness might be part wall thickness. However, part wall thickness does not represent the cavity openness very well. An example is the part shown in Fig. 2 where the part is flat and the thickness in Z direction (the direction coming out from page) is the same. The wall thickness will be the same everywhere but the openness is different in different regions. For example, the openness at B is larger than that at C. However, the variation in openness of different parts of the same shape with differing thickness would be captured by wall thickness.

To calculate the flow resistance, we need to consider both the local openness and flow vector at the same time. The resistance thus is defined as reverse of the multiplication of part voxel number along flow vector and part voxel number at the plane perpendicular to flow vector. This can be shown by the example in Fig. 4 to calculate the flow resistance at flow front voxel A with a flow vector. A flow line is first constructed along flow vector (shown as dash line in Fig. 4) and we count the number of part voxel of local region at this flow line, denoting the number as *m*. Then a flow plane through voxel A but perpendicular to the flow line is constructed, shown as an ellipse in Fig. 4. We count the number of part voxel of local region at the flow plane, denoting as *n*. Note that voxel B is not counted because it is not in local region even it is at the flow plane. Then the flow resistance is ready to be computed as $r = 1/(m \times n)$. By this definition, we can address both local openness and flow vector. The flow resistance then can be used to compute flow speed.

Actually, we only need the reverse of flow resistance, $m \times n$ during the calculation. We define the flow potential (not to be confused with potential energy) as $p = s \times m \times n$, where s is the speed. In each calculation step, we first calculate p for every flow front voxel then take the overall average potential $\overline{p}$ as the threshold. If the potential of a flow front is larger than the threshold, this voxel is said to have enough potential to move. If not, this voxel is held into the next step until it has larger potential than the threshold. In addition, since there are only limited flow vector in discrete space, we can pre-compute the locations of voxels for each flow line and each flow plane associated with each flow vector store them in a table. When computing flow resistance, we only need to check directly if the voxels in the table are cavity voxels. Further, we only check voxels in the flow plane within a certain range, say, 10×10 voxels around the flow front voxel to save computation.

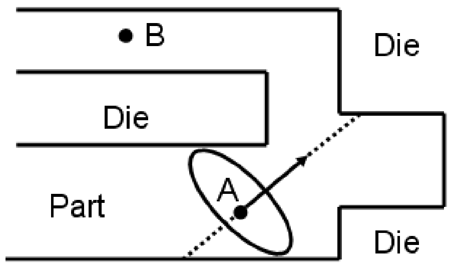


Fig. 4 Calculation of flow resistance

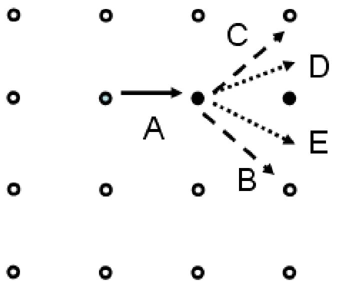


Fig. 5 New vector calculation

3. Vector change at obstruction

When a flow hits an obstruction, the flow vector is going to change. In the previous algorithm, the new flow vector was calculated by simply constructing a vector from the current flow front voxel to the empty voxel. With the improvement, this is changed to the sum of original vector and position vector (vector from current voxel to empty voxel). In Fig 5, solid dots represent obstructions and blank dot represent empty voxels. A flow front voxel has the vector A and hits an obstruction. In the old algorithm, there were two new vectors, B and C while in the new one, the two new vectors are D (sum of A and C) and E (sum of A and B). By this way, new direction is not only related to position but also related to original direction, which gives a better way to calculate new direction. The calculation results using the old and new algorithms are shown in Fig. 6 ~ 7 using a plate part with a through hole, where the hole serves as the obstruction. When the flow hits the obstruction, it should disperse more like the pattern shown in Fig. 7 than Fig. 6.

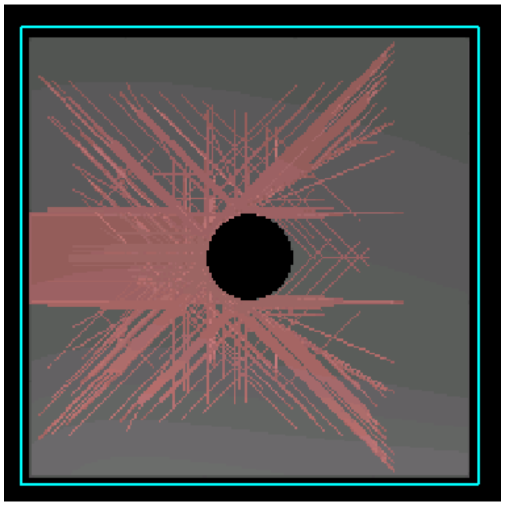


Fig. 6 Flow pattern using old algorithm

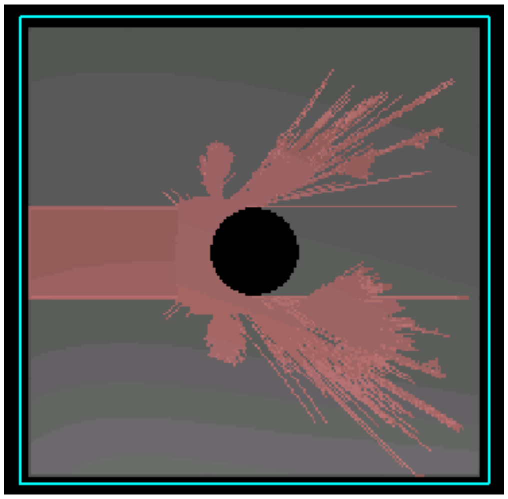


Fig. 7 Flow pattern using new algorithm

There is a pre-computed angle table to search empty voxels when a flow hits obstructions. The table contains voxel locations of 13 different angles, which is between current flow vector and vector to empty voxels, ranging from 35^o to 180^o. The neighboring empty voxels can be found through this angle table. The angle search starts from smallest angle to largest angle. Whenever there is an available angle, which means there is an empty voxel, the search stops and takes this empty voxel to calculate the new vector in the old algorithm. In another word, the search only selected one angel as the output. As a result, some available vectors are ignored and the new vectors are not completed. Fig. 8 shows a fan gate to be filled. The flow should move smoothly from back to front and the flow front should be nearly a “flat line”, which is the design intent of the fan gate. However, as shown in Fig. 9, the result from original algorithm is far from anticipated.

The reason for this is that only one angle was considered to calculate new vectors. As the flow advances, the thickness is smaller and smaller then the resistance becomes larger and larger. The flow would move to any possible direction. If there is one angle limitation, the flow front becomes irregular, as shown in Fig. 9. In the new algorithm, more available angles are provided to be searched. The results for the fan gate for 2 angles and 3 angles searching are shown in Fig. 10 ~ 11. It can be seen that result from 2 angles searching is much better than one angle searching while result of 3 angels searching is better than 2 angles searching. However, there is efficiency penalty to search more angles. Compared to the original run time, the run time on 2, 3 and 4 angels searching are about 25%, 40% and 50% more, respectively. For the tradeoff, we choose 2 angels searching since it is much better than one angle searching and only slightly increases the run time.

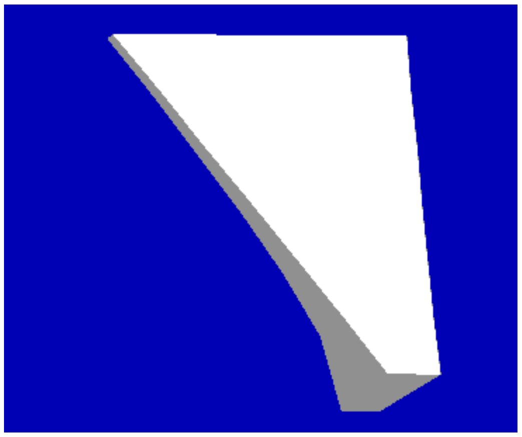


Fig. 8 A fan gate to be filled

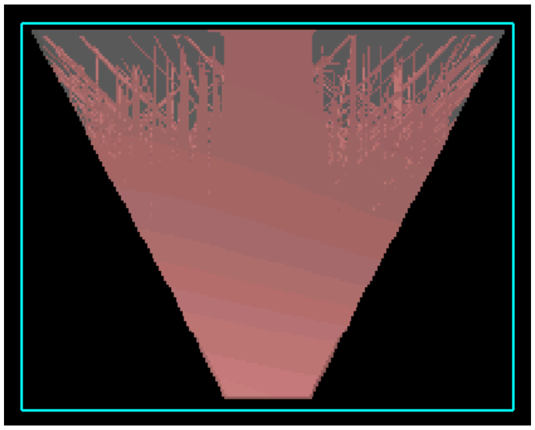


Fig. 9 Result from original algorithm

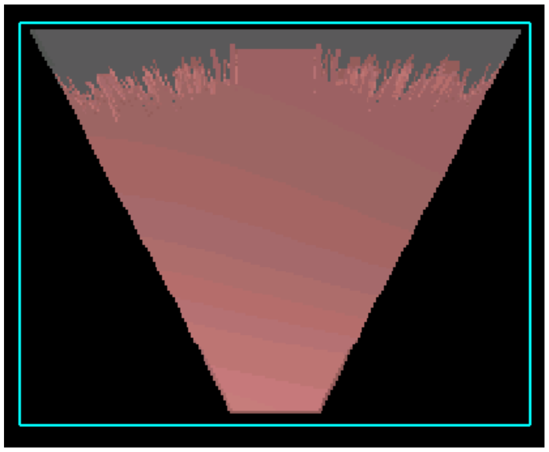


Fig. 10 Result of 2 angels searching

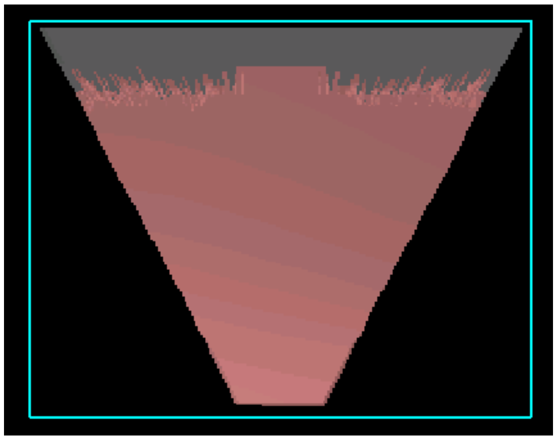


Fig. 11 Result of 3 angles searching

Flow speed is also affected when flow hits obstruction since there is energy loss. The change of speed is related to the angle between the original vector and the new vector. The larger the angle is, the more the speed decreases. For example, the speed loss for the 180^o angle is larger than that for the 35^o angle.

4. Multiple gates

If a practical part has multiple gates with different distances to the biscuit and a properly designed runner system, the flow will start to fill cavity simultaneously due to the large resistance at gates. For the example shown in Fig. 12, cavity fill will start after the runner system pressurizes which should not happen until the runner system is nearly full. Even if melt reaches gate C before the other two, the cavity will start to fill only to a small degree because the gate resistance is significantly higher than the runner resistance. However, in the original algorithm, due to different fill distance of OA, OB and OC, the melt reaches C earlier than it reaches A and B. The melt will then fill the cavity at C before it reaches A and B, which will makes an incorrect prediction for the fill pattern. The flow resistance discussed earlier cannot handle this case because resistance computed at C is not significantly higher than those at A and B. Thus this is an exception for flow resistance calculation.

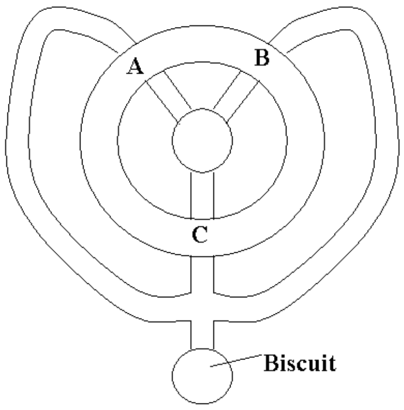


Fig. 12 A part with multiple gates

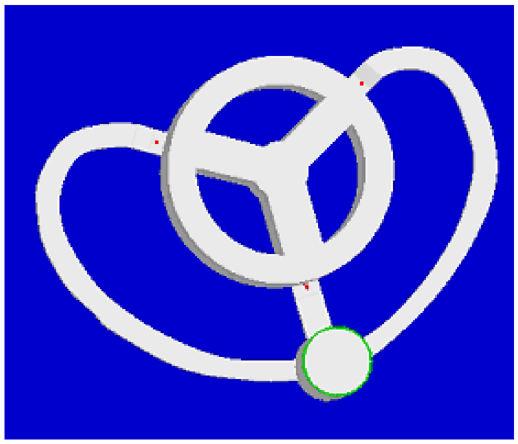


Fig. 13 Three gates are specified

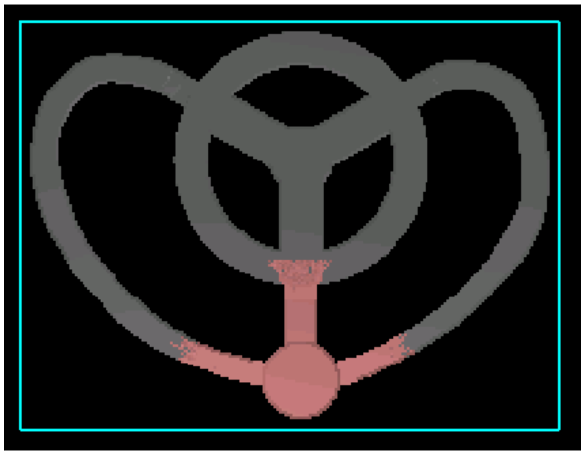


Fig. 14 Result from original algorithm

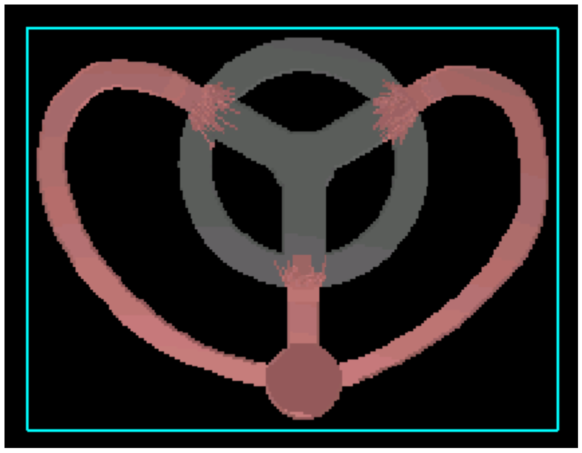


Fig. 15 Result from mew algorithm

We made a change for this exception and provide a function to allow the user to specify multiple gates. In the example of Fig. 13, three gates (identified with a red dot) and the biscuit (in green circle) are specified. During the fill pattern calculation, the surface analysis is done first to find the wall thickness information. For each gate point, the local region with the same thickness is found and is set to be a gate region. All gate voxels are assigned a very large flow resistance, which will force the flow to stop at gate region until the runner system is filled. The flow can start to fill cavity simultaneously from gate regions. Fig. 14 shows the result from the original algorithm while Fig. 15 shows the result after the improvement. It is clearly shown that in Fig 15, the three flows start to fill cavity almost simultaneously.

5. Bias removal

The fill pattern for symmetric part should also be symmetric (for example, the flat part with hole in Fig. 6). However, the analysis result in old algorithm was not symmetric. Careful inspection found that it was caused by calculation sequence. As mentioned earlier, there is a list to store the flow front voxels. At each calculation step, the new flow front voxels are generated based on the sequence of old flow front voxels, i.e. from the front to the end. Since the number of voxel can be filled in a step is fixed due to mass conservation, there is fill competition for flow front voxels, which may generate fill bias. For example of Fig. 6, if in the initial flow front list, voxels at left side are the front and voxels at right side are the end, the voxels at left side will always be filled earlier than those at right side. As the calculation continues, there is more and more bias. The solution is to reverse the sequence at each step. At an odd step, the calculation is from the front of flow front list to its end while at an even step, it is from the end to the front. Thus the bias can be removed.

6. Influence of neighboring voxels

In the old algorithm, the calculation for each flow front voxel was independent of its neighboring flow front voxels. There was no influence between neighboring voxels. As a result, the flow lines were usually very dispersed. This differs from numerical simulation. Furthermore, occasionally there were small flows advancing too far compared to others, which should be incorrect. An example of old algorithm was shown in Fig. 6. Even after we apply the new calculation of vector (shown in Fig. 7), the flow lines are still too dispersed. In the new algorithm, the influence from neighboring voxels is addressed by grouping flow front voxels and balancing their vector and speed, which helps reduce the dispersal. The flow front voxels are first grouped based on their neighbor relationship. In each group, the average vector and speed are calculated. Each flow front voxel is then slightly adjusted to the average vector and speed. This way cannot eliminate the problem completely but can help balance the behavior of voxels within neighborhood.

7. Efficiency issue

Special attention was paid to efficiency for implementation of new algorithm. It is undesired that run time increases significantly due to the added computation. Some methods are applied to reduce computation: 1) Pre-computed tables of geometry information are used. 2) Real data are rounded to integers. 3) Variables are saved for later use. 4) Data structure and computation are optimized. With these treatments, the run time increase for the new algorithm is 20% ~ 50% compared to the old.

ANALYSIS RESULTS

The algorithm was implemented in the software CastView and the fill pattern analysis results of CastView are compared to the numerical simulation results. The first part for test is a simple part. The result from MagmaSoft is shown in Fig. 16 ~ 17 while result from the modified CastView is shown in Fig. 18 ~ 19. From the comparison, it can be seen that both results are quite similar. The flow goes into the central region of part and fills the two far corners and then goes back. The two near corners are the last regions to be filled.

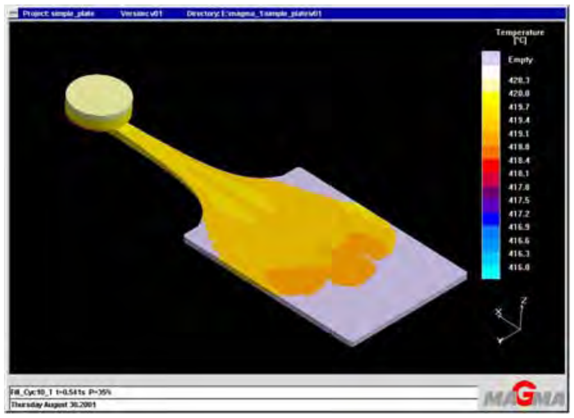


Fig. 16 Result from MagmaSoft (1)

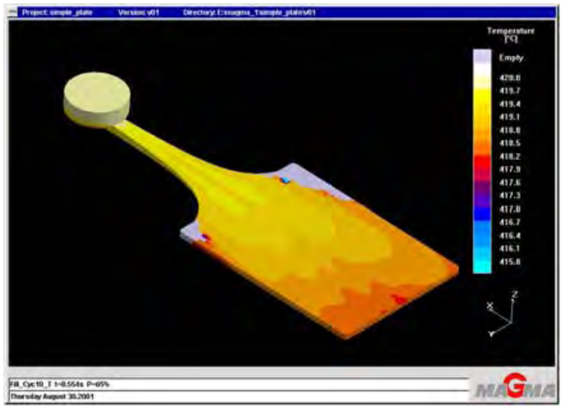


Fig. 17 Result from MagmaSoft (2)

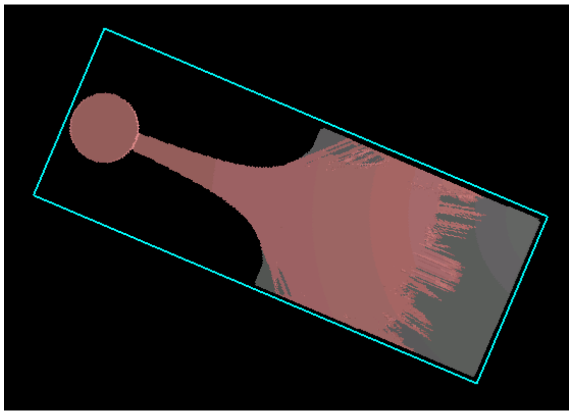


Fig. 18 Result from CastView (1)

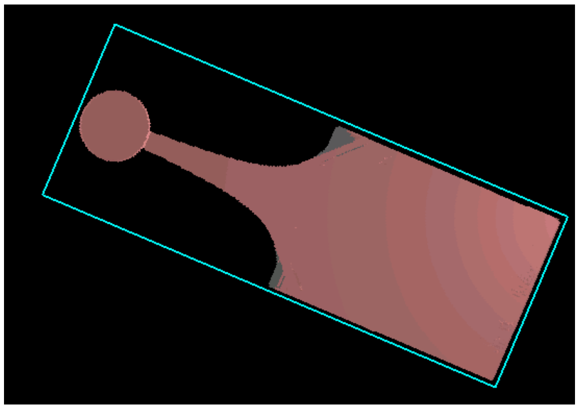
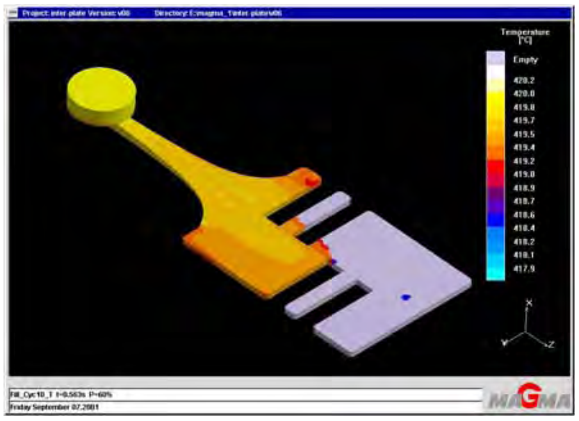


Fig. 19 Result from CastView (2)

The second part for fill pattern is a finger part. The numerical simulation result using MagmaSoft is shown in Fig. 20 ~ 21. It can be seen that the first finger is filled simultaneously with the second half of part and the second finger is the last region to be filled. The result from CastView is shown in Fig. 22 ~ 23. We can see that the pattern is similar to that of numerical simulation. The first finger is not filled until the second half of part is filled. The second finger is the last region to be filled.



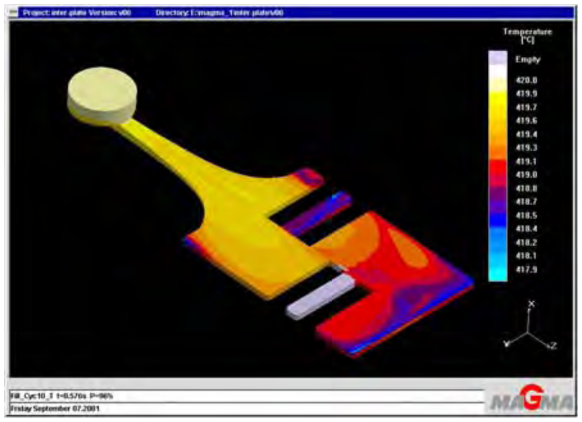
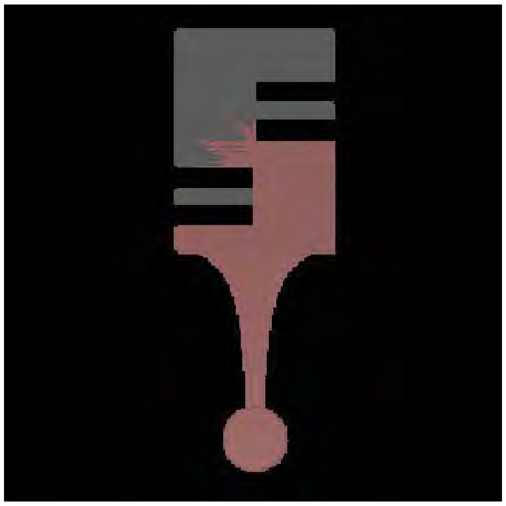


Fig. 20 Result from MagmaSoft (1) *Fig. 21 Result from MagmaSoft (2)*



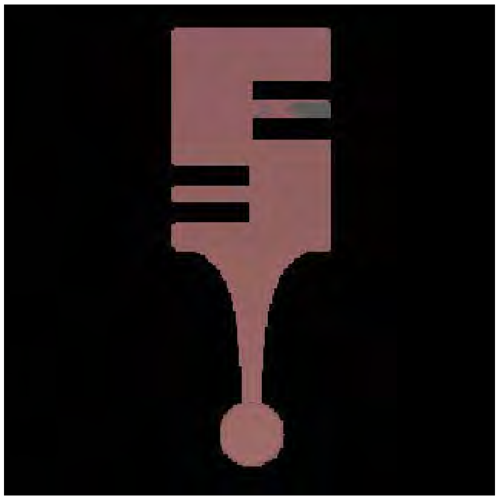


Fig. 22 Result from CastView (1) *Fig. 23 Result from CastView (2)*

FUTURE WORK

Though improvement was made in the current algorithm, there is still much work to be done in future to further improve the analysis accuracy. This may include:

1. Redefinition of flow resistance. There are special cases of geometry that the current definition of flow resistance cannot measure correctly. Since the flow resistance plays an important role in flow front voxels calculation, it is necessary to seek a better definition to address the complex local geometry.
2. Vector calculation when hitting obstruction. Though the calculation in current algorithm is apparently better than that in old algorithm, sometimes it has problems in that it generates some incorrect vectors. As a result, some cavity regions are filled earlier than they should be.
3. Effect of obstruction and flow resistance on speed change. Currently, the flow speed is changed in a simple way using angle change and flow resistance, which is insufficient for the speed calculation. A better method should be applied for the calculation of speed.
4. Computational efficiency. More focus will be placed on efficiency to reduce the run time.
5. More tests and comparison. This is a necessary step to verify the algorithm. The analysis results from fill pattern reasoning will be compared to those from numerical simulation which has better accuracy.

CONCLUSIONS

Geometric reasoning is an alternative way for fill pattern analysis of die casting process in addition to numerical simulation. It can provide quick results for the fill pattern without requiring much input information, which is particularly important at the early design stage and for tasks such as quoting. The original algorithm implemented in CastView has been improved in some areas:

1. Speed calculation is included. The speed of flow front voxels is calculated in each step based on flow resistance, obstruction and a parenting relationship.
2. Flow resistance is defined. The flow resistance is defined based on the local geometry and flow vector and it determines if a flow has sufficient energy to move.
3. Vector change calculation at obstruction is improved. The new vector is calculated based on not only neighboring empty voxels but also the original vector.
4. Multiple gates can be specified. This ensures the cavity fill can be started at multiple gates simultaneously.
5. Bias in fill pattern due to computation sequence is fixed.
6. Influence of vector and speed from the neighboring flow front voxels is considered.
7. Special attention is paid to computational efficiency. Comparison of the results from geometric reasoning to results from numerical simulation suggests its efficiency and validity.

REFERENCES

[1] Li, Y. B. and Zhou, W., Numerical Simulation of Filling Process in Die Casting, Materials Technology, Vol. 18, No. 1, 2003, pp. 36-41.
[2] Jia, Liangrong; Xiong, Shoumei and Liu, Baicheng Study on numerical simulation of mold filling and heat transfer in die casting process Journal of Materials Science and Technology, Vol.16, No. 3, 2000, pp. 269-272.
[3] McLaughlin, Martin; Kin, Chung-Whee and Backer, Gerald, The Importance of Trapped Gas Analysis in Magnesium Filling Simulations, Die Casting Engineer. Vol. 47, No. 3, 2003, pp. 32-36.
[4] Heine, R. W. and Uicker, J. J. Risering by Computer Assisted Geometric Modeling, Transactions of the American Foundrymen's Society, Vol. 91, 1983, pp.127-136.
[5]. Ma, Y., Qualitative Geometric Reasoning for Thermal Design Evaluation of Die Casting Dies, Ph.D. Dissertation, 2000, The Ohio State University.
[6] Elfakharany, Essameldin F., Qualitative reasoning for filling pattern in high-pressure die-casting and gravity-driven casting, PhD dissertation, 1999, The Ohio State University.

Appendix B: Visualization of Slow Fill Processes

Visualization of Slow Fill Casting Processes

Dongtao Wang
Center for Die Casting
Ohio State University

September 2004

1. Introduction

A common concern of casting designers is the fill pattern of metal liquid as it enters the mold cavity. Filling is one of the most important factors to determine the product quality. The metal flow, if not controlled precisely, can cause flow-related defects, which range from gas porosity and knit lines to incomplete-fill. The prevailing method of predicting the cavity fill pattern is numerical simulation using one of the commercially available systems available for this purpose. However, due to the complexity of the phenomena and the equations used to model the process, fill simulations can be time consuming to setup and time consuming to run. Furthermore, to use them properly the simulation systems require that the user have experience with the process and a good understanding of fluid dynamics. These disadvantages limit the utility of numerical simulation and the number of "what-if" questions, especially early in design.

An alternative method is qualitative reasoning that provides significantly less information but also requires much less of the user. This method is based on the geometric characteristics of the mold cavity so it only requires the input of the part and gate geometry and does not require fluid dynamics background of the user. There are no complex equations to be solved so the prediction can be obtained quickly making it particularly suitable for the early stage design where maximizing the number of "what-ifs" is important.

Several years ago the CastView research group at The Ohio State University developed a qualitative reasoning algorithm for fill pattern visualization in high pressure die casting (Miller, 1998, Rebello, 1997 and Elfakharany, 1999) that has proven to be helpful for gating system design and vent placement. The fill pattern calculation in CastView is limited to the high pressure die casting process and is not applicable for other processes, such as gravity die casting or squeeze casting. Study of these processes shows that the major difference between them is dominating force that controls the filling processes. By accounting for the relative contributions of the forces that control filling it is possible to modify the geometric reasoning approach used for die casting fill to other casting processes.

From the fluid dynamics point of view, the filling characteristics of gravity casting, low pressure or permanent mold die casting, squeeze casting and semi-solid casting can be captured by the relative magnitude of the forces involved. For example, in die casting, the viscous term and body force term usually are both negligible due to its small viscosity and high velocity. However, for gravity casting, the body force term is the most important term and for semi-solid casting, the viscous term is important but inertial term only plays minor role due to the large viscosity and low velocity. Nevertheless, it is still possible to apply the similar model to calculate the fill patterns for those different filling processes with only relatively minor modification is required during the implementation.

In this research, the fill model for die casting process was modified to calculate/visualize the fill pattern of those slow fill processes. The purpose is to build a “general” algorithm of fill pattern analysis for different processes that can be applied with minor modifications in parameters. Two slow fill processes , gravity casting and squeeze casting, are considered.

2. Consideration from Navier-Stokes Equations

The well-known Navier-Stokes equations are widely applied to solve fluid flow problems in numerical simulation. Starting from Navier-Stokes equations in this research provides some useful hints about the dominant terms in our modeling.

The Navier-Stokes equations in rectangular coordinates are written as:

$$\begin{aligned}
\rho(\frac{\partial v_x}{\partial t}+v_x\frac{\partial v_x}{\partial x}+v_y\frac{\partial v_x}{\partial y}+v_z\frac{\partial v_x}{\partial z}) &= -\frac{\partial p}{\partial x}+\mu(\frac{\partial^2 v_x}{\partial x^2}+\frac{\partial^2 v_x}{\partial y^2}+\frac{\partial^2 v_x}{\partial z^2})+\rho b_x \\
\rho(\frac{\partial v_y}{\partial t}+v_x\frac{\partial v_y}{\partial x}+v_y\frac{\partial v_y}{\partial y}+v_z\frac{\partial v_y}{\partial z}) &= -\frac{\partial p}{\partial y}+\mu(\frac{\partial^2 v_y}{\partial x^2}+\frac{\partial^2 v_y}{\partial y^2}+\frac{\partial^2 v_y}{\partial z^2})+\rho b_y \\
\rho(\frac{\partial v_z}{\partial t}+v_x\frac{\partial v_z}{\partial x}+v_y\frac{\partial v_z}{\partial y}+v_z\frac{\partial v_z}{\partial z}) &= -\frac{\partial p}{\partial z}+\mu(\frac{\partial^2 v_z}{\partial x^2}+\frac{\partial^2 v_z}{\partial y^2}+\frac{\partial^2 v_z}{\partial z^2})+\rho b_z
\end{aligned} \qquad (1)$$

Since the terms in the equations have dimensions, it is not easy to see the dominant terms (or the important terms). Some algebra of scaling helps with this decision. To do the scaling analysis, we need to define the dimensionless variables as follows:

$$t^* = \frac{t}{t_c}$$

$$x^* = \frac{x}{L}$$

$$y^* = \frac{y}{L}$$

$$z^* = \frac{z}{L}$$

$$v_x^* = \frac{v_x}{V}$$

$$v_y^* = \frac{v_y}{V}$$

$$v_z^* = \frac{v_z}{V}$$

$$p^* = \frac{p}{\Delta p}$$

$$b_x^* = \frac{b_x}{B}$$

$$b_y^* = \frac{b_y}{B}$$

$$b_z^* = \frac{b_z}{B}$$

where:

t_c --- cycle time;

L --- characteristic dimension. It can be the length of part;

V --- characteristic velocity; It can be the gate velocity;

Δp --- characteristic pressure gradient.

B -- gravity acceleration;

The Navier-Stokes equations now can be changed to dimensionless form. Making use of the chain rule for differentiation:

$$\frac{\partial}{\partial t} = \frac{dt^*}{dt}\frac{\partial}{\partial t^*} = \frac{1}{t_c}\frac{\partial}{\partial t^*}$$

and similarly:

$$\frac{\partial}{\partial x} = \frac{1}{L}\frac{\partial}{\partial x^*}$$

$$\frac{\partial}{\partial y} = \frac{1}{L}\frac{\partial}{\partial y^*}$$

$$\frac{\partial}{\partial z} = \frac{1}{L}\frac{\partial}{\partial z^*}$$

$$\partial v_x = V\partial v_x^*$$

$$\partial v_y = V\partial v_y^*$$

$$\partial v_z = V\partial v_z^*$$

Using these relations to introduce the dimensionless variables into the governing equations, with $b_x^* = b_y^* = 0$ (supposing z direction is pointing up) yields:

$$\left[\frac{\rho V}{t_c}\right]\left(\frac{\partial v_x^*}{\partial t^*}\right)+\left[\frac{\rho V^2}{L}\right]\left(v_x^*\frac{\partial v_x^*}{\partial x^*}+v_y^*\frac{\partial v_x^*}{\partial y^*}+v_z^*\frac{\partial v_x^*}{\partial z^*}\right)=-\left(\frac{\Delta p}{L}\right)\left(\frac{\partial p^*}{\partial x^*}\right)+\left[\frac{\mu V}{L^2}\right]\left(\frac{\partial^2 v_x^*}{\partial x^{*2}}+\frac{\partial^2 v_x^*}{\partial y^{*2}}+\frac{\partial^2 v_x^*}{\partial z^{*2}}\right)$$
$$\left[\frac{\rho V}{t_c}\right]\left(\frac{\partial v_y^*}{\partial t^*}\right)+\left[\frac{\rho V^2}{L}\right]\left(v_x^*\frac{\partial v_y^*}{\partial x^*}+v_y^*\frac{\partial v_y^*}{\partial y^*}+v_z^*\frac{\partial v_y^*}{\partial z^*}\right)=-\left(\frac{\Delta p}{L}\right)\left(\frac{\partial p^*}{\partial y^*}\right)+\left[\frac{\mu V}{L^2}\right]\left(\frac{\partial^2 v_y^*}{\partial x^{*2}}+\frac{\partial^2 v_y^*}{\partial y^{*2}}+\frac{\partial^2 v_y^*}{\partial z^{*2}}\right)$$
$$\left[\frac{\rho V}{t_c}\right]\left(\frac{\partial v_z^*}{\partial t^*}\right)+\left[\frac{\rho V^2}{L}\right]\left(v_x^*\frac{\partial v_z^*}{\partial x^*}+v_y^*\frac{\partial v_z^*}{\partial y^*}+v_z^*\frac{\partial v_z^*}{\partial z^*}\right)=-\left(\frac{\Delta p}{L}\right)\left(\frac{\partial p^*}{\partial z^*}\right)+\left[\frac{\mu V}{L^2}\right]\left(\frac{\partial^2 v_z^*}{\partial x^{*2}}+\frac{\partial^2 v_z^*}{\partial y^{*2}}+\frac{\partial^2 v_z^*}{\partial z^{*2}}\right)+\left[\rho B\right]b_z^*$$
(2)

All terms in brackets are dimensionless. Thus we only need to compare the coefficients in the square brackets. The values typical for die casting of aluminum alloy,

$\mu = 3\times10^{-3}\,Pa\ s$ --- Viscosity
$\rho = 2.7\times10^{3}\,kg/m^3$ --- Density
$V = 10m/s$ --- Speed
$L = 0.1m$ --- Fill distance
$B = 9.8m/s^2$ --- Z component of Gravity

Substituting these values into Eq. (1-4) thru (1-6) we have for

High Pressure Die Casting:

Viscous term: $\frac{\mu V}{L^2}$ = $3\times10^{-3}\times10$ / 0.1^2 = 3.
Inertial term: $\frac{\rho V^2}{L}$ = $2.7\times10^3\times10^2$ / 0.1 = 2.7×10^6.
Gravity term: ρB = $2.7\times10^3\times9.8$ = 2.6×10^4.

These data show the viscous term is negligible in die casting and the gravity term is also negligible compared to inertial term. Clearly, the dominant term is the inertial term. For gravity casting and squeeze casting, the speed is 0.5 *m/s* and 1 *m/s* respectively and the corresponding data is (assuming the material properties remain the same):

Gravity casting:

Viscous term: $3\times10^{-3}\times0.5$ / 0.1^2 = 0.15.
Inertial term: $2.7\times10^3\times0.5^2$ / 0.1 = 6.8×10^3.
Gravity term: $2.7\times10^3\times9.8$ = 2.6×10^4.

Squeeze casting:

Viscous term: $3\times10^{-3}\times1 / 0.1^2 = 0.3$.
Inertial term: $2.7\times10^3\times1^2 / 0.1 = 2.7\times10^4$.
Gravity term: $2.7\times10^3\times9.8 = 2.6\times10^4$.

It can be seen that for gravity casting, the dominant force is gravity but the inertial term also has some effect. The viscous term is still negligible. For squeeze casting, both the inertial term and gravity term are dominant terms.

Considering the above typical values, the order of processes for increasing importance of gravity is: die casting, squeeze casting and gravity casting. The gravity effect in die casting is while for gravity casting, the momentum effect is nearly negligible but gravity term is the dominant term.

To model the fill pattern of gravity casting and squeeze casting, two component vectors are applied to represent the effects of inertial term and gravity term, namely a momentum vector and a potential vector. The momentum vector depends on the flow velocity while the potential vector depends on the flow location. The combination of these two vectors determines the flow vector. In die casting, momentum vector is the dominant vector since the filling process is under high pressure with high velocity while in gravity casting, the gravity vector is the dominant vector since the velocity is small. If both flows are dominating (for squeeze casting), the actual flow is the sum of both flows.

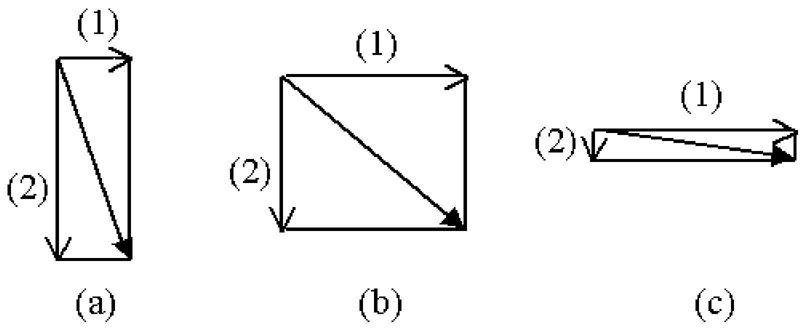


Figure 1 Relation between momentum vector and potential vector.

Vectors (1) and (2) represent momentum vector and potential vector, respectively. Cases (a), (b) and (c) represent gravity casting, squeeze casting and die casting (Elfakharany, 1999).

The calculation suggests that it is possible to visualize the fill pattern for slow processes using a single algorithm. The major difference is to determine the dominant flow or to calculate the contribution of different flows. Then the resistance and speed for each flow front voxel can be calculated and the fill pattern analysis can be performed.

3. Fill Pattern for Gravity Casting

The previous analysis based on dimensionless Navier-Stokes equations proves the dominant term in gravity casting is the gravity term, while the inertial term takes some effect.

Elfakharany developed an algorithm for fill pattern reasoning for gravity casting. His model is based on the analysis of Bernoulli's equation. See **Error! Reference source not found.** for an illustration of the terms in the equation. The Bernoulli's equation states that the sum of potential head, the velocity head and the pressure head of a flowing liquid is constant and is in this form:

$$\frac{V_1^2}{2g}+\frac{P_1}{\rho}+h_1=\frac{V_2^2}{2g}+\frac{P_2}{\rho}+h_2 \tag{3}$$

Where:

V_1 = metal velocity at point 1.
g = acceleration due to gravity.
P_1 = static pressure in the liquid at point 1;
h_1 = height of liquid at point 1.
ρ = density of the liquid.

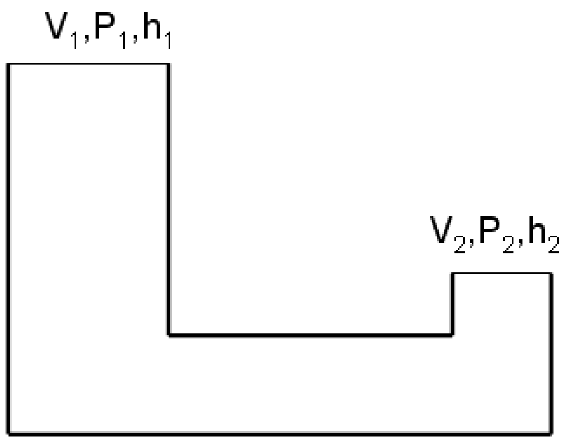


Figure 2 Illustration of Bernoulli's Principle

Bernoulli’s equation is an energy balance equation, which divides the flow energy at a given point into three parts, kinetic energy, pressure energy and potential energy and the sum of these three energies is constant. Though Bernoulli’s equation only applies in equilibrium, Elfakharany uses the concept to define the momentum vector and potential vector.

In Elfakharany’s model, the sum of momentum value and potential value is always equal to a constant. The potential vector is always pointing in the –Z direction and the value is relative to the height (Z coordinate value) of current flow front voxel. If the maximum Z value of voxel model is *n*, a flow front voxel with Z coordinate of *n* has the maximum potential value. Similarly, a flow front voxel with Z coordinate of 0 has minimum potential value. The direction of the momentum vector is the current flow front vector. During the computation, since the Z coordinate value is known, the value of potential can be calculated. Since the sum of potential and momentum values is known, the value of momentum vector can be calculated. Given the potential direction and momentum direction are known, the sum of two vectors can be calculated, which is the new flow front direction.

Elfakharany’s model provides a basic idea to calculate the effect of gravity vector. However, As discussed above, the dominant term in gravity casting is the gravity term. The ratio of importance between gravity term and inertial term is about 10:3. However, in Elfakharany’s model, the ratio used to calculate new vector was 1:1 which is not correct.

The new algorithm was implemented and integrated into CastView and the flow chart is shown in Figure 3.

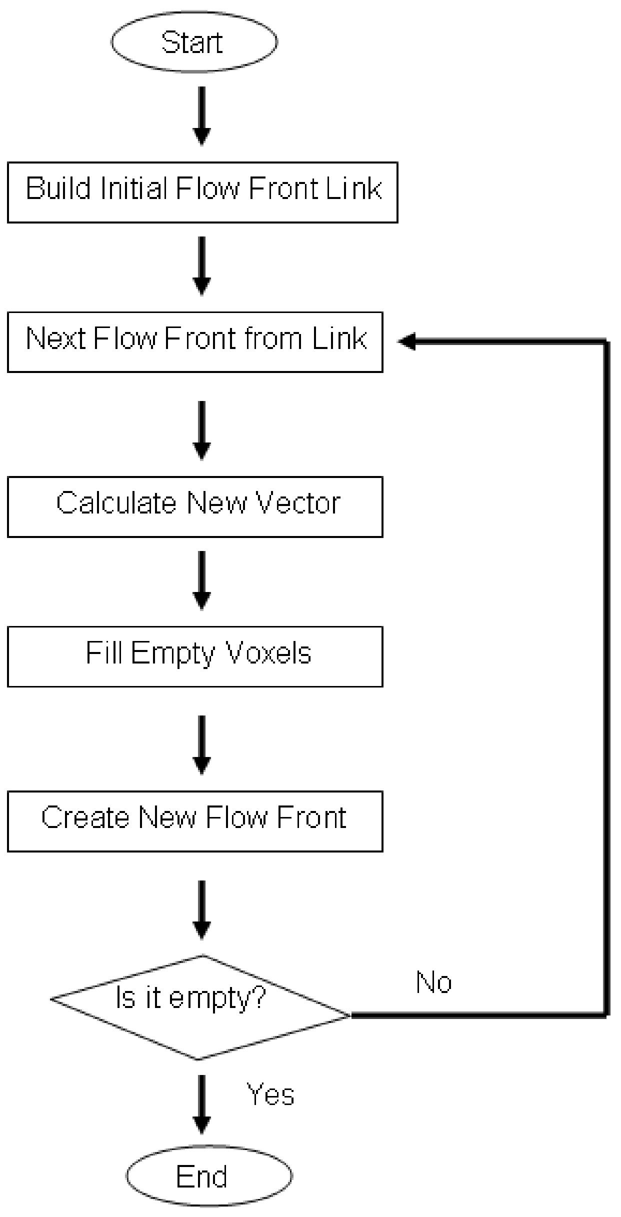


Figure 3 Calculating fill pattern of gravity casting based on combination of die casting fill pattern algorithm and Elfakharany's algorithm

3.1 Case Study: Gravity Casting

The example is taken from the paper "Simulation of Die Filling in Gravity Die Casting Using SPH and MagmaSoft" by Ha, et. al. (1999). The paper includes die model dimension, experiment results, MagmaSoft results and SPH results. This provides a way to compare CastView results with results from other approaches.

Figure 4 shows the cavity geometry and gate as well as runner system. The water analogue modeling was also performed as experiment. Ha and his associates built a transparent die to record the filling of water.

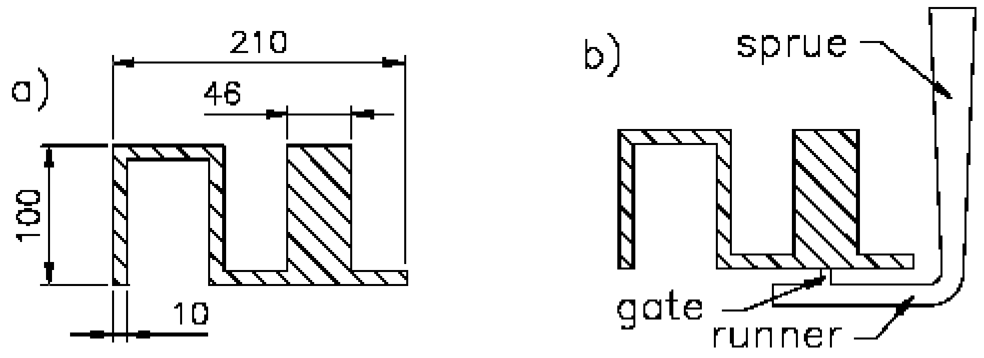


Figure 4 Model for experiment and simulation.

The lengths are in mm and the third dimension is 20 mm thick. (Ha, et. al. 1999)

A CAD model was built using Unigraphics NX and the STL model was exported for CastView analysis. The rendering of part in Unigraphics and STL are shown in

Figure 5 and Figure 6. The experiment results and numerical simulation results are shown in Figure 7. The left result is from MagmaSoft while middle result is from experiment and right result is from SPH simulation. It can be seen that the flow fills the wide rib first then fill the narrow rib along the channel. Both simulation results match the experiment very well.

The geometric reasoning is performed on this model for fill pattern by Elfakharany's algorithm and the new algorithm. Results from both algorithms are listed in Figure 8.

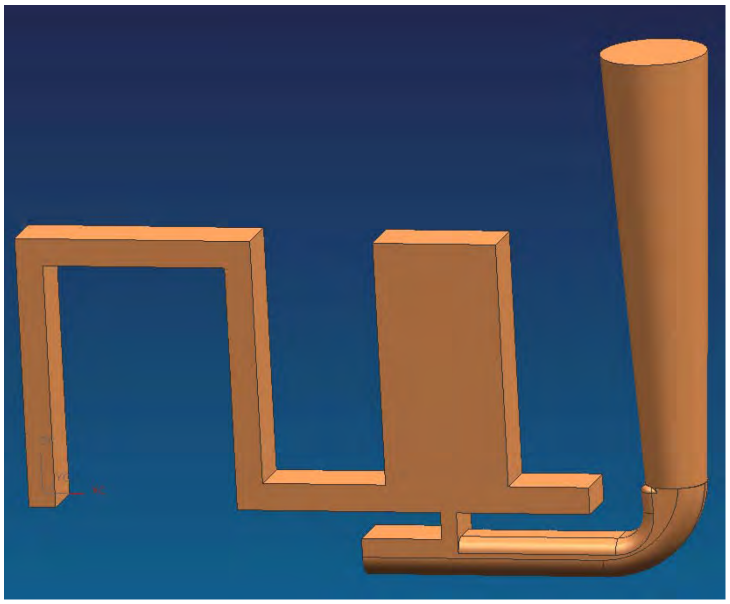


Figure 5 Die cavity to be tested on CastView, shown in Unigraphics

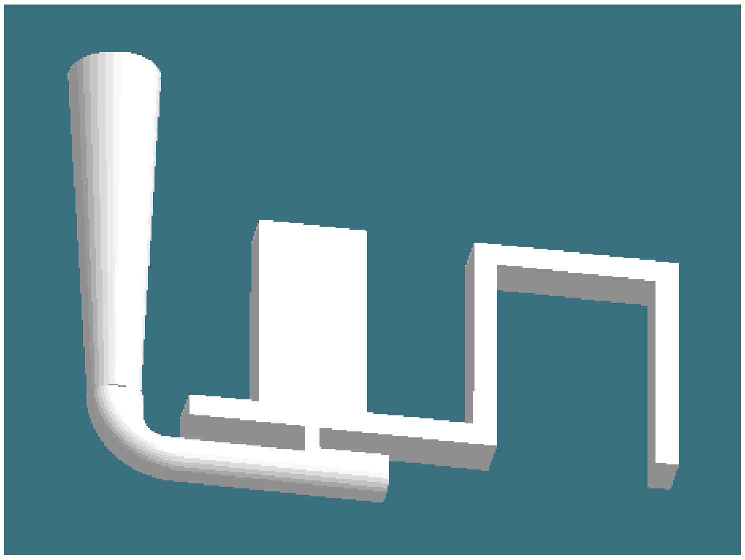


Figure 6 Die cavity in STL shown in CastView

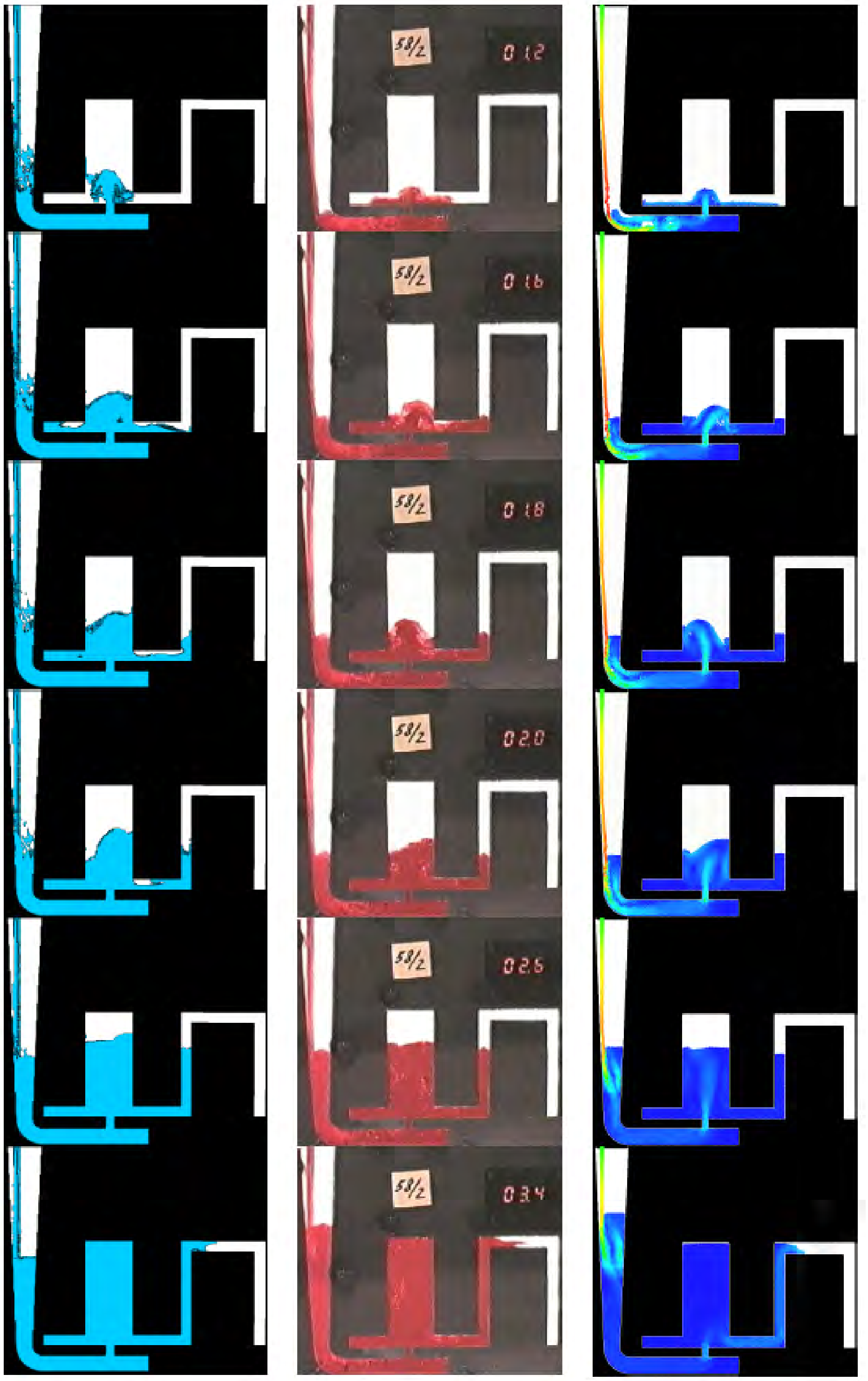


Figure 7 Numerical simulation and experiment results. (Ha, et. al. 1999)

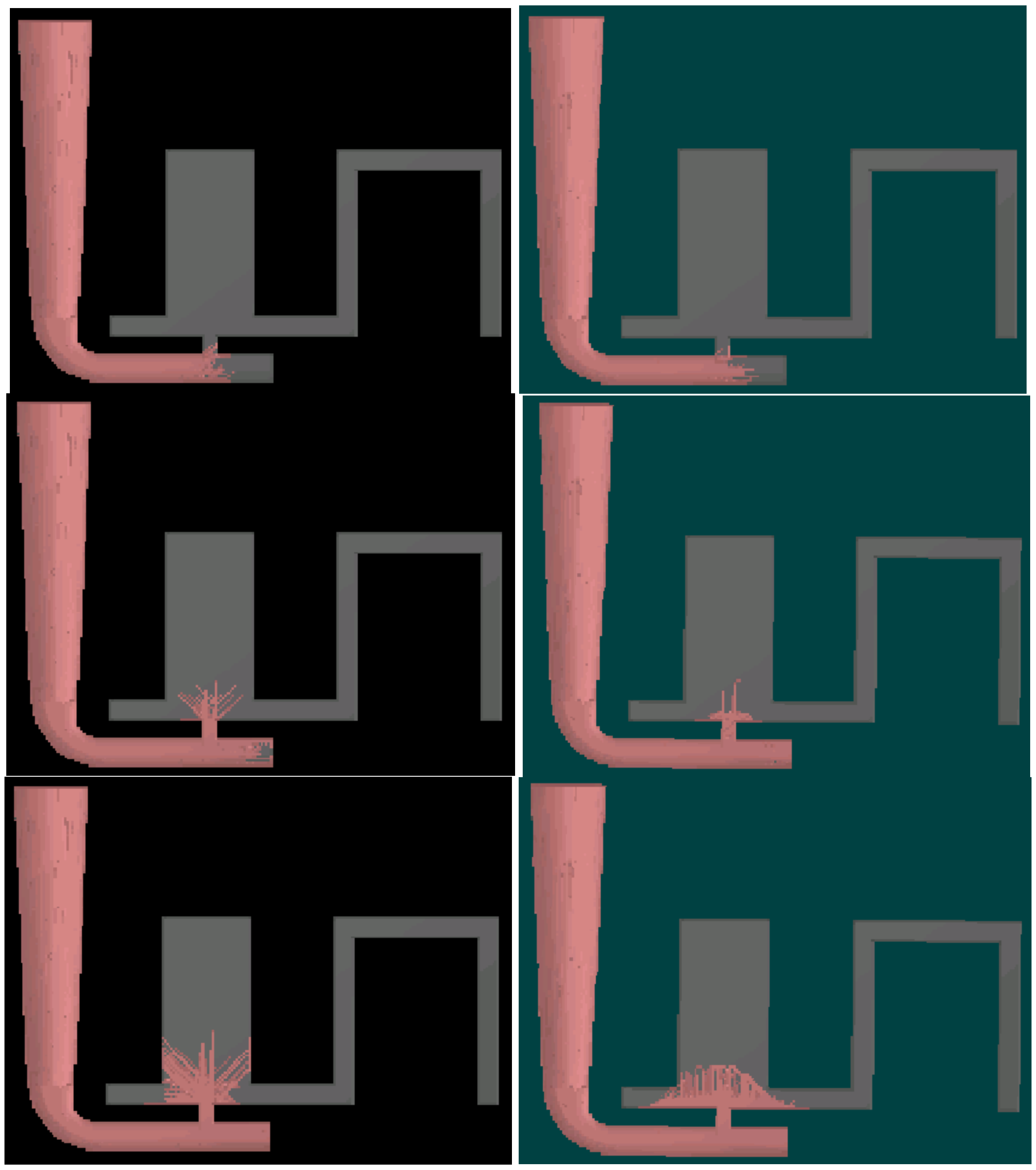


Figure 8 CastView results of fill pattern for the same gravity casting part. Left: Elfakharany's algorithm; Right: new algorithm

Comparing the geometric reasoning results from both algorithms and those from Ha's paper, we can see that the results match quite well. The flow fills the wide rib then the narrow rib, which demonstrates the suitability of geometric reasoning for gravity casting. However, comparison of the details shows some difference. When the flow is entering the wide rib, it forms a fountain due to gravity effect. This fountain can be seen at the results of numerical simulation and experiment but not in the result from Elfakharany's algorithm. However, this fountain can be clearly seen in the results from the new algorithm. That is because in the new algorithm, the weight on gravity is larger based on the similarity results described above. Another improvement is the pattern after the wide rib has been completely filled and the flow starts to fill the horizontal narrow rib. The results of simulation and experiment show that the left end of horizontal rib is filled before flow reaches the right end, which exactly matches the result from new algorithm. In results from Elfakharany's algorithm, the flow starts to fill the final vertical rib before the left end of horizontal rib is filled, which is not correct. This comparison shows the improvement of fill pattern for gravity casting.

3.2 Fill Pattern for Squeeze Casting

As discussed previously, the dominant terms for die casting, gravity casting and squeeze casting are different, which can be seen from the dimensionless Navier-Stokes equations. In die casting process, the dominant term is the inertial term and the gravity term is negligible. In gravity casting, the dominant term is gravity term and inertial term only plays a minor role for the fill pattern. However, in squeeze casting, there is not a single dominant term. Gravity term and inertial term play almost equal roles in determining the fill pattern.

In our algorithm, we choose the ratio of gravity term and inertial term as 1:1 to calculate the vector for the new flow front. The algorithm is implemented in CastView based on the fill pattern algorithm for the die casting process.

Two examples were chosen for test. The experiment and numerical simulation are from the paper by Wallace, et. al. (2000). Both parts are flat plates with different thickness. One is in 0.5 inch and the other one is in 0.25 inch. The dimensions of the two parts for squeeze casting are shown in Figure 9. The flow pattern of the molten metal into the die cavity was measured by the use of incomplete shots and the numerical simulation was performed using Procast.

The CAD models of these two parts are built on Unigraphics using the dimensions in Figure 9. The 3D rendering pictures in Unigraphics are shown in Figure 10.

The experimental results and Procast results of 0.5 inch plate are listed in Figure 11. It can be seen that at the early stage when melt fills the cavity, the flow advances like an arc with some bias at one side due to gravity effect. The flow front then becomes even and advances steadily until it reaches the end.

The CastView results are shown in Figure 12. When the flow starts to enter the cavity, it forms a big bump around the gate with some bias at one side, which is very similar to the pattern in Figure 11. The flow then advances more evenly and steadily to the end. It can be seen that the results in Figure 11 and Figure 12 have good agreement, which indicates the suitability of fill pattern reasoning for squeeze casting.

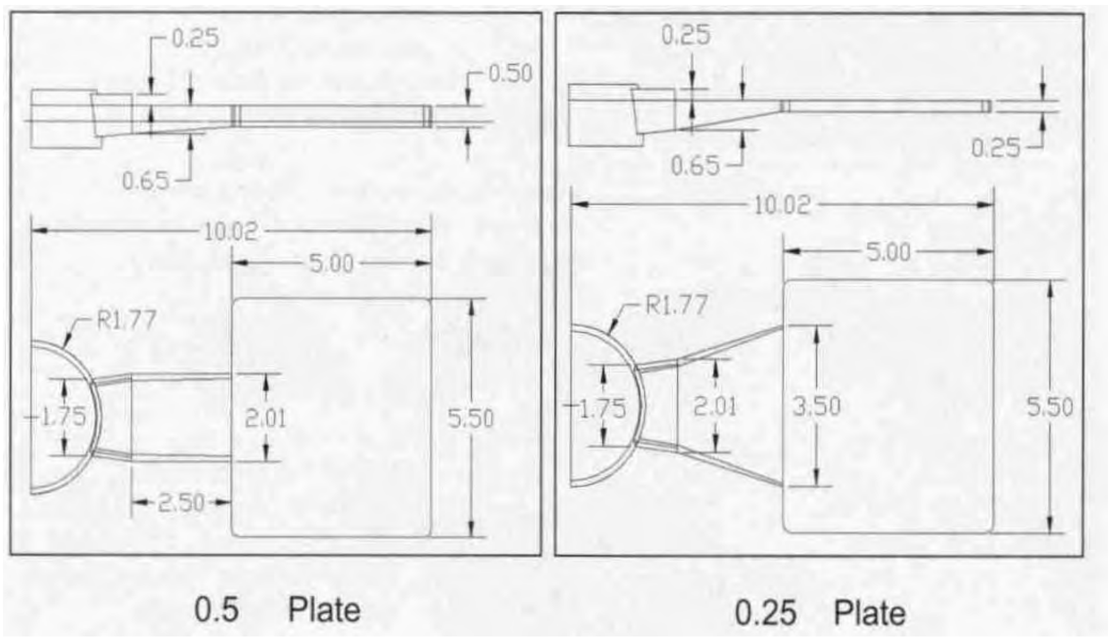
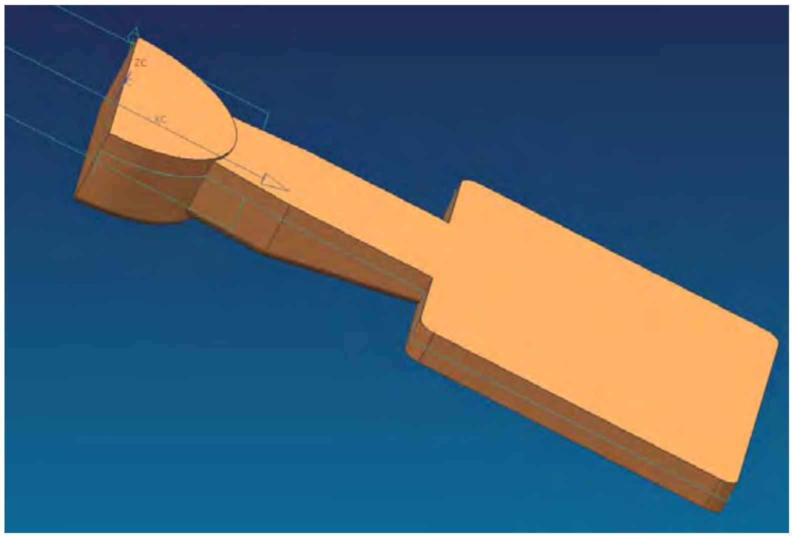


Figure 9 Dimensions of two plates for squeeze casting. The difference of the two parts is the plate thickness and gate shape. (Wallace, et. al. 2000)



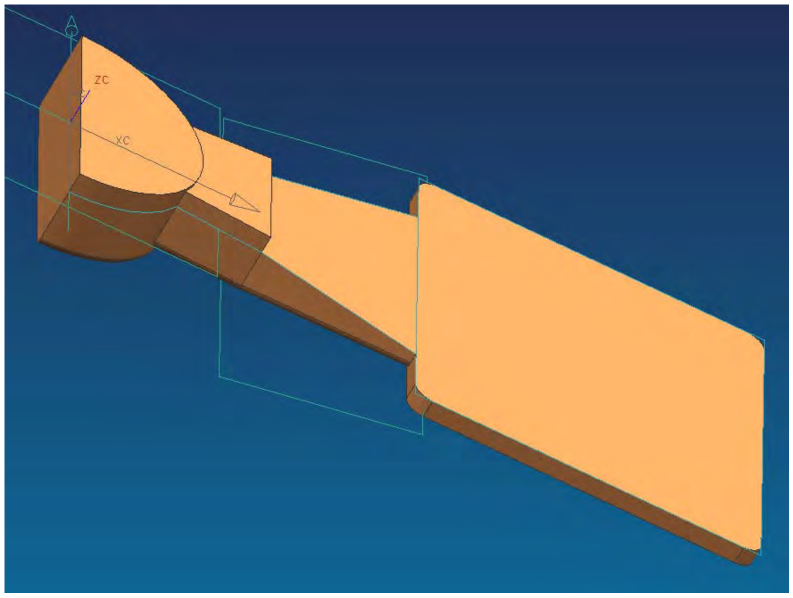


Figure 10 Two plates with different thickness for squeeze casting. Upper: 0.5 inch plate; lower: 0.25 inch plate

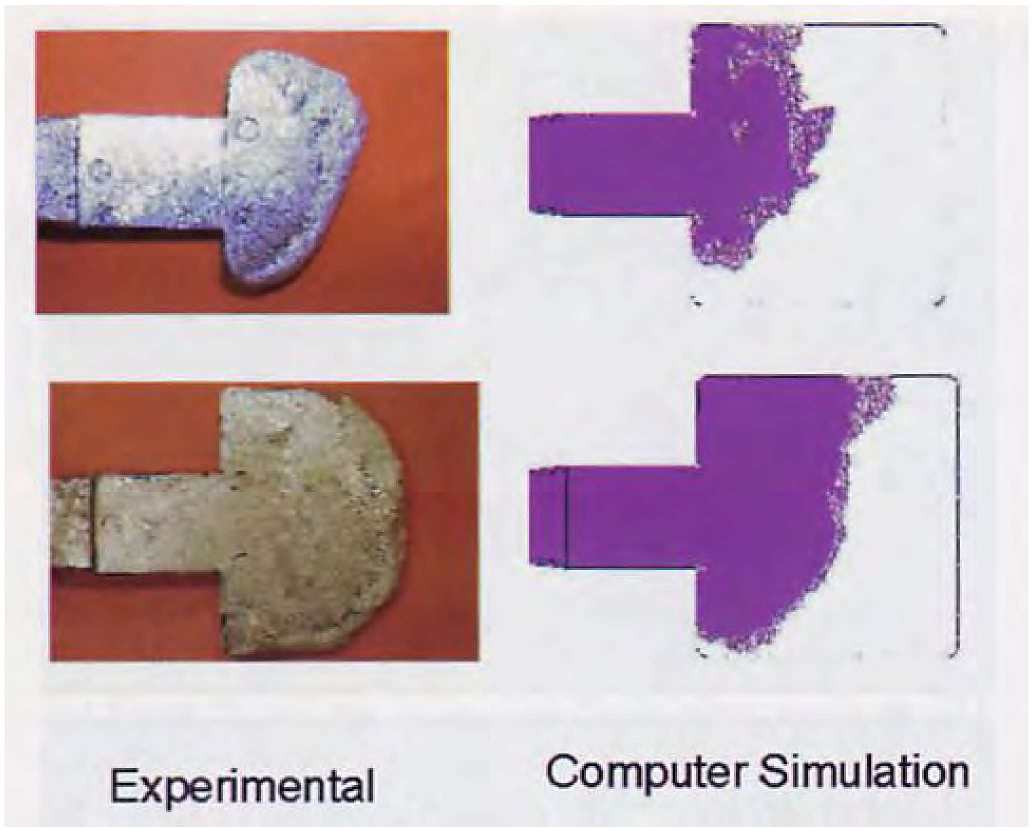
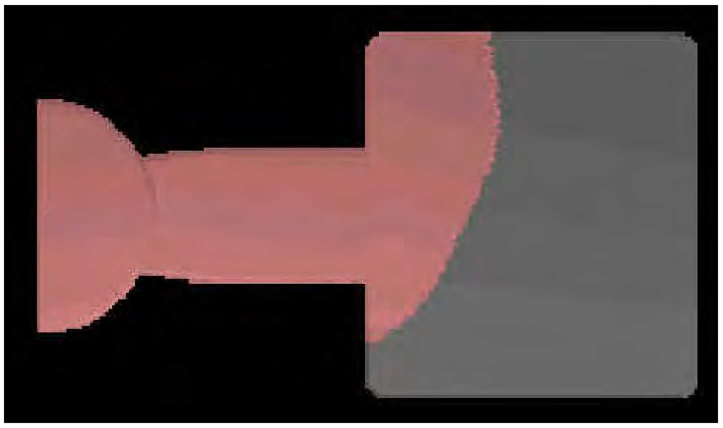


Figure 11 Experimental results and simulation results from Procast for 0.5 inch plate. (Wallace, et. al. 2000)



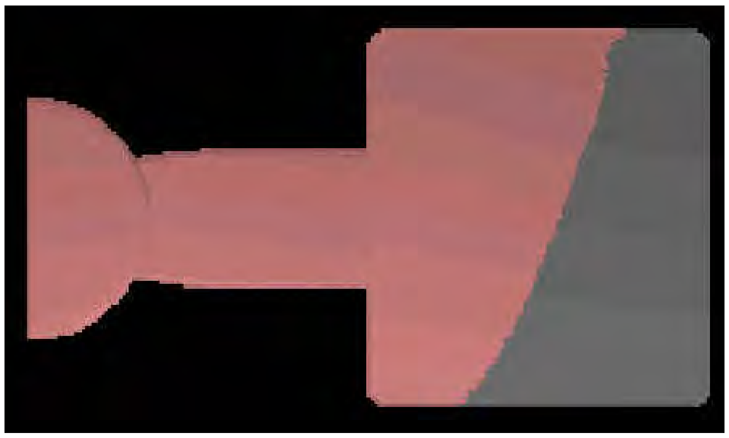


Figure 12 CastView result for 0.5 inch plate. At the beginning of cavity fill, the flow forms a bump from gate. The flow front then becomes flat and steady

Another example is the 0.25 inch flat part. The paper only provides the simulation result for the 0.25 inch flat part without experimental result. Figure 13 shows the numerical simulation result of the thin plate by Procast. The flow fills the fan gate then starts to fill the cavity in a much more uniform front. The flow front then advances steadily to fill the whole cavity. The corresponding results from CastView for the same part are shown in Figure 14. It can be seen that once the flow enters the cavity, the flow front is in a uniform form. The flow front then fills the cavity steadily from left to right until it hits the end line. Though there is still a little disturbance at the flow corner, the overall flow pattern from CastView is very similar to that from Procast. The paper does not provide the run time for the numerical simulation by Procast but a reasonable guess is 2 ~ 3 hours on a typical PC. Given the analysis time for this part by CastView is 2 ~ 3 minutes.

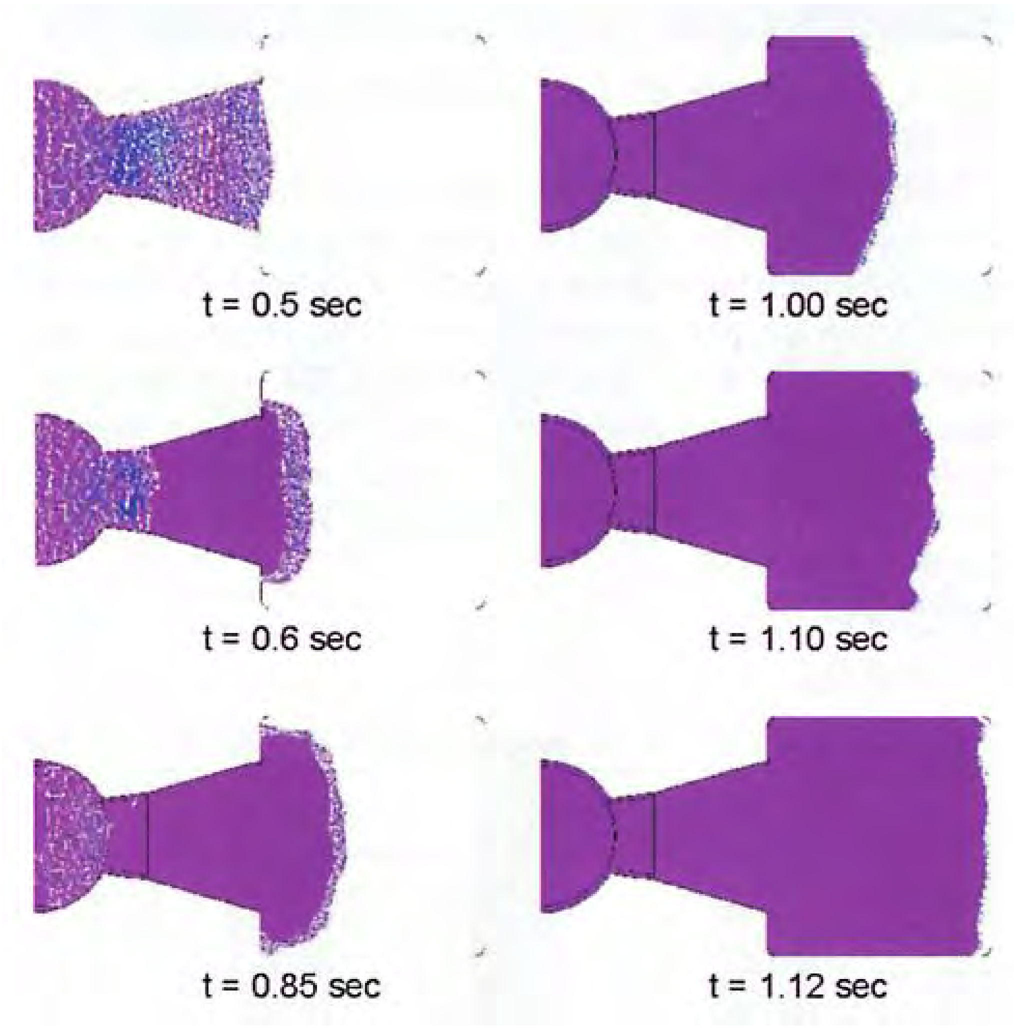


Figure 13 Fill pattern by numerical simulation using Procast for 0.25 inch plate. The flow is in a uniform front to fill the whole cavity. (Wallace, et. al. 2000)

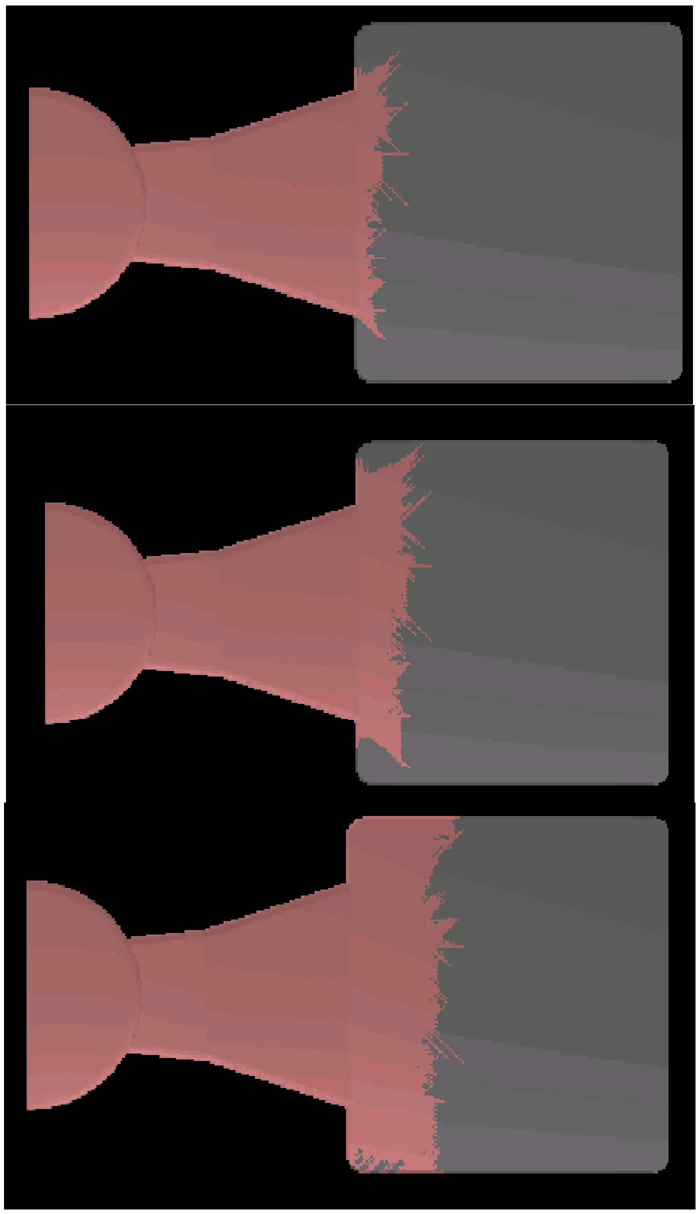


Figure 14 Result of fill pattern for 0.25 inch plate from CastView. The flow front is uniform and advances steadily until it hits the end

4. Conclusions

1. It has been shown that the fill pattern algorithm for die casting can be applied to slow fill processes with slight modification.
2. It can be seen from the Navier-Stokes equations that the major difference between die casting, gravity casting and squeeze casting is the dominant term in flow calculation. The algebra further shows that the dominant term is the combination of gravity term and inertial term.
3. There were problems with Elfakharany's algorithm for gravity casting that were easily corrected. This algorithm was redesigned and implemented based on the algorithm for die casing process.
4. A gravity casting case was chosen to test the algorithm. Analysis was performed using the old algorithm and new algorithm and the results compared to experiment and numerical simulation results. The comparison clearly shows the validity and efficiency of improvement in new algorithm.
5. Two squeeze casting examples were constructed with comparisons made between short shot results, numerical simulation results and CastView results. The fill patterns are very similar, which indicates the efficiency of applying geometric reasoning technique on slow fill processes.

5. References

1. Elfakharany, Essameldin F., Qualitative reasoning for filling pattern in high-pressure die-casting and gravity-driven casting, PhD dissertation, The Ohio State University, 1999.

2. Ha, Joseph; Cleary, Paul; Alguine, Vladimir and Nguyen, Thang, Simulation of die filling in gravity die casting using SPH and MagmaSoft, *Proceedings of 2nd International Conference on CFD in the Minerales and Process Industries*, 1999.

3. Miller, R. Allen, Lu, Shao-Chiung and Alexander B. Rebello, Simple visualization techniques for die casting part and die design, Final report, DOE project # DE-FC07-95ID13362, 1998.

4. Rebello, A. B., Visualization of the filling of die-casting dies, PhD dissertation, The Ohio State University, 1997.

5. Wallace, John F.; Chang, Qingming and Schwam, David, Process Control in Squeeze Casting, *Die Casting Engineer*, 2000, November/December, p42-48.

Appendix C: Modeling Dynamic Cavity Pressure and Impact Spike in Die Casting

Modeling Dynamic Cavity Pressure and Impact Spike in Die Casting

H. Xue
K. Kabiri-Bamoradian
R. A. Miller
The Ohio State University, Columbus, OH, USA

Copyright 2005 American Foundry Society

ABSTRACT

The cavity pressure is difficult to measure in die casting due to the harsh environment inside the cavity, the high temperature and pressure, and solidification of molten metal. Even if it were practical to measure pressure, it would be extremely difficult, if not impossible, to measure the spatial distribution of the pressure throughout the cavity. The prediction of defect distribution in the cavity, and ultimately the casting microstructure requires a good understanding of the pressure distribution. Prediction of die distortion and prediction of flash due to the impact spike will be improved with a better characterization of the pressure distribution. Computer simulations will be extremely useful for these problems if the history of the cavity pressure distribution can be predicted to an acceptable level. The purpose of this research is to develop a dynamic model of the transient cavity pressure that can be used in die distortion modeling and also to gain insight about the influence of the shot profiles on the cavity pressure. An experimental cast part was modeled in order to conduct the cavity filling simulation. The simulation model consisted of shot sleeve, plunger, runner and casting. As expected, the simulation results indicate a high-pressure spike develops at the end of the die-cavity filling phase.

INTRODUCTION

In the high-pressure die-casting process, a plunger is used to rapidly push molten metal into a die cavity. The molten metal, the plunger, plunger rod, hydraulic piston, and some amount of hydraulic oil are in motion at a high speed during cavity fill and rapidly decelerate at the end of fill. The sudden deceleration and stop of all these moving masses can create a very high dynamic pressure – impact pressure, in the die cavity depending on the masses of the system and deceleration time. The role of cavity pressure and the effects of impact pressure developed at the end of fill have been documented in the literature as discussed below.

Considerable work in predicting die distortion during the casting process using computer modeling techniques has been conducted at the Center for Die Casting at Ohio State University. Computer simulation of the die casting process is a relatively inexpensive way to assist the die and machine designers to improve die and machine performance. Previous modeling of die distortion and deformation has ignored the dynamic effects of the moving masses and the rapid deceleration and treated the problem by analyzing pressure and clamping as static loads. In a static simulation of a die casting process, the cavity pressure is treated as uniformly distributed pressure on the cavity surface and is assumed to remain constant for throughout the cycle [Choudhury, A.K., 1997, Kesavan, V., 1999]. However, in reality the cavity pressure is transient and spatial location dependent, and the die will respond to this dynamic input load. The pressure peak developed at the end of fill, combined with the dynamic responses of the die casting machine and dies, causes the die components to deform and may cause the die to open therefore affecting the onset of flash and possibly the dimensions of the cast part.

The pressure spike in the cavity is caused by the force needed to decelerate the moving masses. The cavity pressure peak is responsible for large deflections of the die which may lead to flashing [Ahuett-Garza, H., Miller, R.A., 1997]. The effect of high pressures on casting quality was investigated and the insights indicated that high impact pressure does not always improve the casting quality as high intensification pressure. The high impact pressure or high plunger velocity makes flashing more likely [Savage, G. et al, 2001]. J. R. Mickowski et al stated that the impact pressure should be controlled such that the plunger velocity and final pressure are retained. He also developed a real-time closed-loop shot control system to control the impact pressure. In this system, a rapid response valve was added to the hydraulic cylinder so that an extremely rapid deceleration in a small final fraction of the plunger stroke can be applied, by which overall cavity fill time at least as short as in the un-controlled conventional process was maintained [Mickowski, J.R. et al, 1993]. Peter Olmsted developed a very innovative idea that uses the overflows to cushion the impact pressure. By adjusting the overflow volume and the entry size to overflows, the impact pressure can be controlled in a very robust way [Branden, J., Olmstead, P. and Kuhn, C.W., 2002].

Limited experimental research in cavity pressure measurements has been done. G. Savage et al put a load cell inside the plunger rod coupling to measure shot rod force during injection process. The backpressure of molten metal and intensification pressure were derived from the shot rod force and plunger tip area. K. Tong et al presented a measurement method in thin-wall magnesium die casting experiments. Quartz force sensors were placed behind the ejector pins, and the metal pressure was calculated from the ejector pin force and its tip area [Tong, K.K.S., et al, 2002]. The methods mentioned above are indirect measurements. Direct measurements were also carried out by G. Savage et al using in-cavity sensors. In order to measure the transient cavity pressure, pressure sensors were placed into the ejector die and aligned to the cavity surface. The wires of sensors came out of cavity through a drilled hole on the ejector die [Venkatasamy, V., Brevick, J., Mobley, C., 1997]. Special pressure sensors were needed to work in the adverse conditions during cavity filling process.

Cavity pressure is very difficult to be accurately measured due to the complexity of die casting process. For instance, the complicated die and casting geometries, severe working conditions during cavity filling process, the fact that cavity pressure can change dramatically in milliseconds, and the solidification of molten metal on the pressure sensors. Therefore computer modeling and simulation techniques were used in this paper to predict the cavity pressure change during the filling.

MODELING APPROACH

The general-purpose CFD program *FLOW-3D* was used as the simulation software. The computation domain includes an experimental cast part (casting), which is used at Center for Die Casting at the Ohio State University, plus shot sleeve, plunger, biscuit, and runner, as shown in Figure 1. Only half of the domain was modeled due to the symmetry of geometries and loading conditions. Multi-block meshing is used to generate a reasonable and sufficient resolution to accurately represent the main geometry and flow features. The current simulation model uses 2 meshing blocks, as shown in Figure 2. The red colored region represents the shot sleeve, and the blue colored region represents the casting. In order to better represent the boundary layer effect at the gate, three cells across the gate section were used.

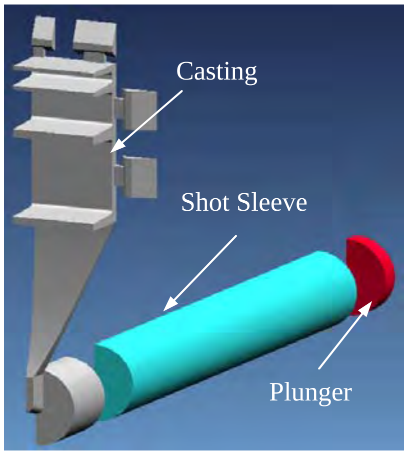


Fig. 1. Simulation model

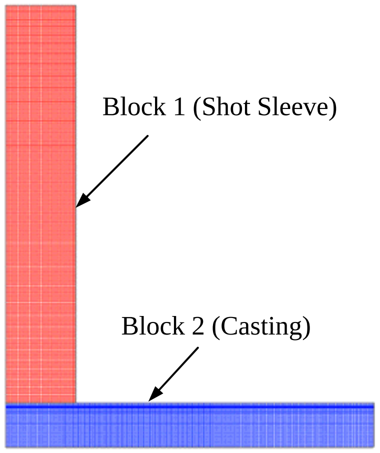


Fig. 2. Two meshing blocks

MODELING ASSUMPTIONS

The molten aluminum is treated as slightly compressible viscous flow with material properties independent of temperature. The temperature loss is very limited during cavity filling. The K-ε turbulent model is used. Heat transfer between dies and molten metal is considered, but no conduction in the dies was included in the simulation. Constant void pressure is specified to represent perfect venting during filling process. The plunger is represented by a moving obstacle with a prescribed velocity profile. At this point, no surface tension and solidification of molten metal are considered.

PLUNGER VELOCITY PROFILE

The molten metal is pushed into the die cavity by a plunger. Figure 3 shows the plunger velocity profile used in the simulation. Based on the NADCA recommendation, the critical slow shot velocity can be obtained from the fill percentage of shot sleeve and plunger diameter. When the plunger is moving at critical slow shot speed the air entrapment due to the plunger motion can be minimized. The calculated critical slow shot speed for the simulation part is 50cm/sec. The fast shot velocity, which is 350 cm/sec, was calculated based on the gate design and required fill time. The transition from slow shot to fast shot was assumed to be a constant acceleration. The deceleration was determined by fast shot velocity and fill time.

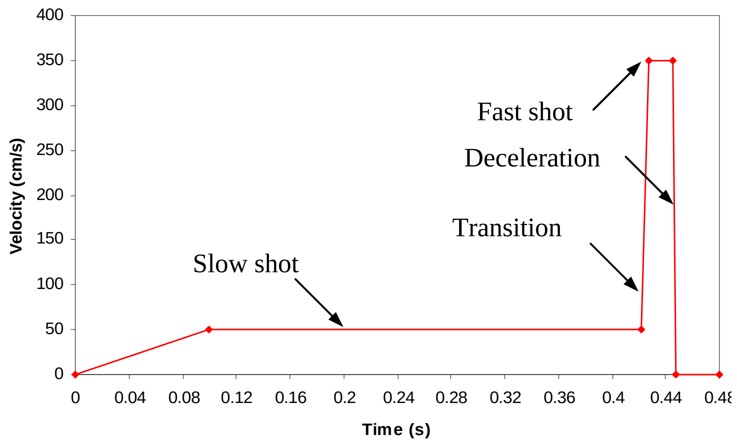


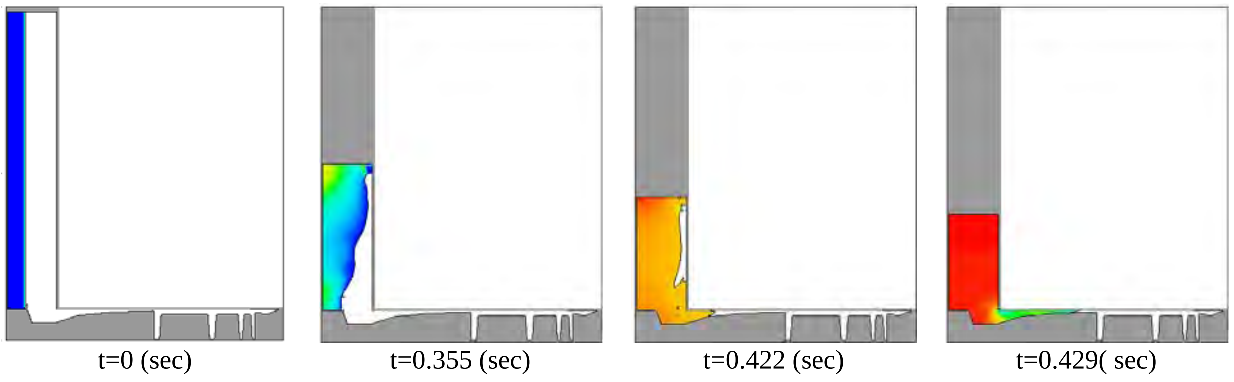
Fig. 3. Plunger shot profile

RESULTS AND DISCUSSION

A cavity filling simulation starting from plunger slow shot through to complete deceleration was conducted. The simulation results are discussed as follows.

FILLING PATTERN

Figure 4 shows the filling pattern at the symmetry plane for different times. It can be seen that during the slow shot, the plunger moves in such a way that the roll of the wave front is prevented and the entrapped air is minimized. However, at the end of slow shot (t=0.422s), some air entrapment was observed, which is caused by the dramatic geometry change when metal flows into the runner. During the fast shot (from t=0.429s), the metal flows filling the upper portion of the cavity first, and then proceeds downwards filling the lower regions of the cavity.



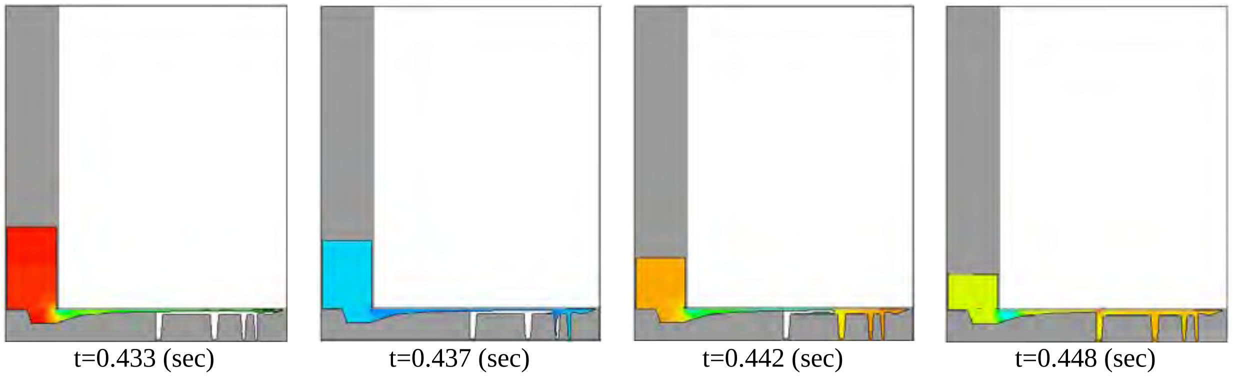


Fig. 4. Filling pattern at the symmetry plane

TRANSIENT CAVITY PRESSURE PROFILE

Figure 5 shows the pressure distribution when the impact occurs. The pressure spike is about 149 MPa (22,000 PSI), which is more than twice of the intensification pressure used in our static simulation for this experimental cast part (10,000 PSI). It was also observed that the pressure within the cavity is very uniform; the maximum pressure difference in the cavity is very small (only 50 PSI).

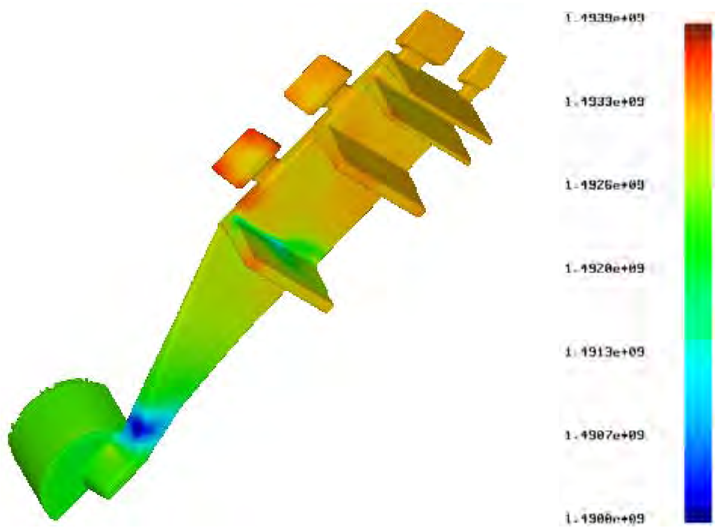


Fig. 5. Impact pressure distributions

The pressure history at different locations is also investigated. The selected inspection locations were shown in Figure 6 and the corresponding pressures were plotted in Figure 7. The blue curve with marks is the velocity profile, and the other curves show the pressure profiles at selected points. It was observed that the metal pressures are very small everywhere due to the perfect venting was assumed. And then a little bit increase during the fast shot. The pressure in shot sleeve is about 500 PSI during the fast shot, which is caused by the resistance below the gate. The close-up pressure profiles during the fast shot are shown in Figure 8. It can be seen that the farther the location is away from the gate, the earlier the pressure starts to increase. This can be explained from the filling pattern: the metal flows to cavity and fills the upper portion of the cavity first, and then the metal flows towards to fill the lower portion. Finally, all the pressures increase in an extremely short time, which indicates the impact occurs and the impact pressure is pretty much uniform.

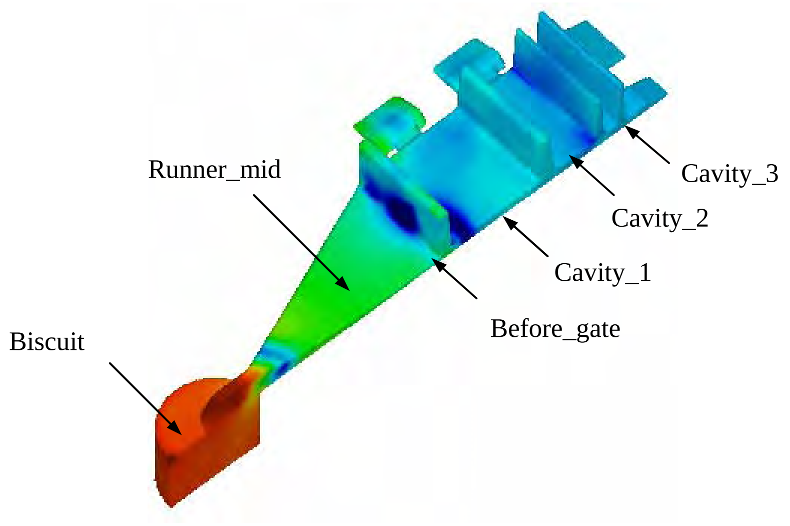


Fig. 6. Inspection locations in casting

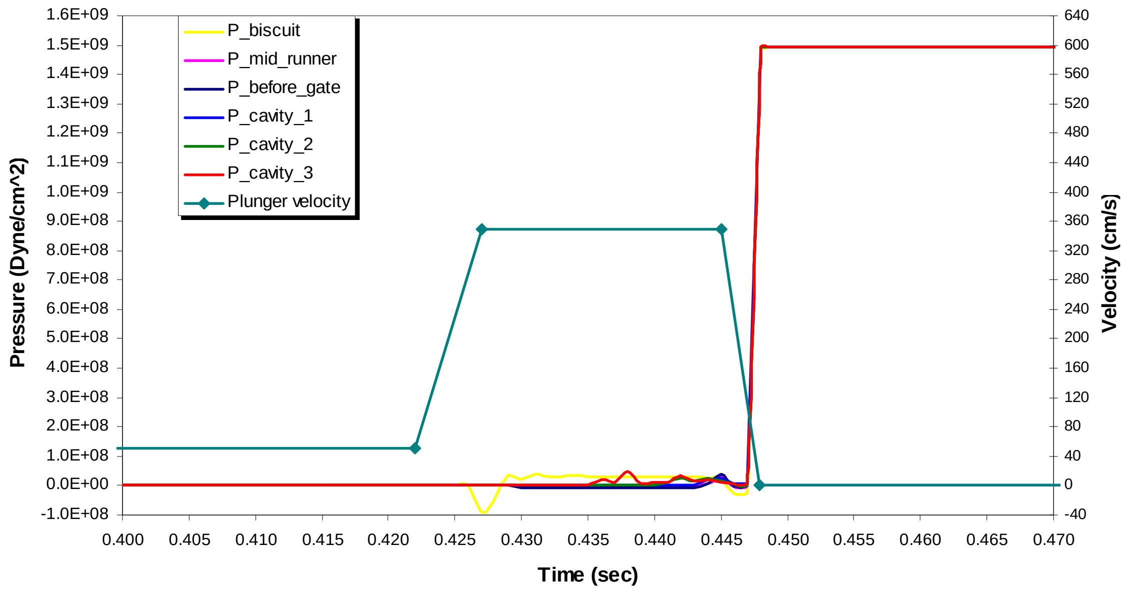


Fig. 7 Pressure profiles at inspection points

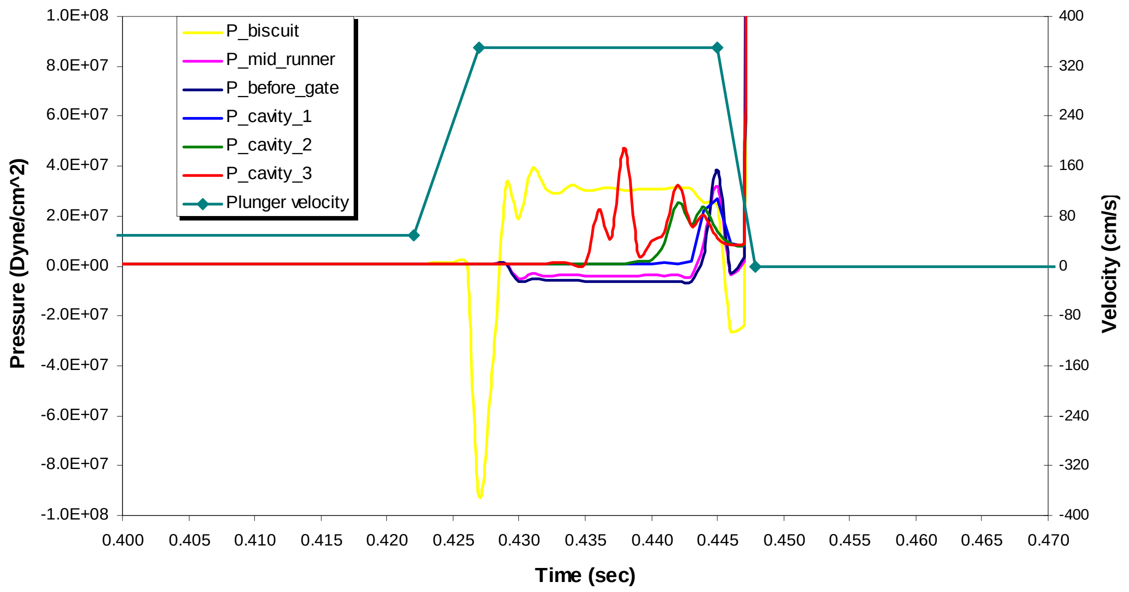


Fig. 8 Pressure profiles at inspection points, close up

DISCUSSION OF FLUID FLOW MODELING

Note that in figure 7, the cavity pressure remains a high value after the plunger velocity becomes zero, which is impossible in an actual injection process. This is because the *Flow-3D* cannot model the interaction between the die and liquid metal. This interaction causes the die to deform and move when impact occurs, which will release the high impact pressure rapidly. In the *Flow-3D* simulation, the die is modeled as a rigid body and cannot move during the shot. This means that the stiffness of the die is infinite and the die can be subjected to the loads without any deformation. The cavity pressure depends on the feeding and cavity geometries and the plunger shot profile derived from the hydraulic system. In an actual die casting process, the impact pressure load is applied to die-cavity surface by liquid metal and causes the die to deform. The volume of the die-cavity will increase due to the die deformation and the molten metal will follow the deformed die-cavity shape. Therefore the cavity pressure is dissipated rapidly by the deformation of the die. An example is die flashing. When die flashing happens, the fluid volume inside the cavity increases and the cavity pressure decreases. As a result, the impact pressure is released rapidly. However, in a *Flow-3D* simulation, the impact pressure peak will remain unless a negative velocity is assigned to the plunger in order to simulate the impact pressure release similar to cavity expansion effect.

The molten metal is always assumed as either incompressible or slightly compressible fluid due to its very large bulk modulus. Due to variations in either shot volume or shot profile, if the plunger keeps moving forward once the cavity is filled, it is possible that a small amount of molten metal is compressed into the cavity. In this case as predicted by the definition of fluid compressibility, $dp = B*dV/V$ (B – bulk modulus, V – total volume of molten metal), the pressure impact is very sensitive to the volume change. Therefore numerical simulation limitation may predict very large impact pressure if the numerical volume of molten metal changes between time steps.

During the filling simulation, if a pressure boundary condition is specified at the plunger, the final cavity pressure will simply be equal to that pressure plus dynamic pressure. However, when a velocity boundary condition is specified at the plunger, the final cavity pressure could be infinite large if the shot profile is not accurate enough as discussed above.

CONCLUSIONS AND FUTURE WORK

In this paper, the cavity filling and injection process in the die casting was simulated by fluid flow modeling using CFD software. Simulation results show that the transient cavity pressure can be predicted and pressure impact is observed, but the results are only as good as the velocity profile that is used as input. Since the pressure field and plunger velocity are tightly coupled and the pressure simulation is computed from the velocity (uncoupled), accurate pressure results at the end of fill and the onset of the impact spike are not possible using such simulations. The plunger deceleration profile used in the simulation should be determined by the fluid flow simulation software rather than defining it as a boundary condition.

Achieving this objective requires coupling the velocity and pressure calculations and ultimately will require relaxing the assumption that the die is a rigid body. Opportunities exist for better modeling the pressure impact by developing coupled models.

Ongoing activities in this research include:

- Customizing calculations with *FLOW-3D* according to the force balance on shot system, i.e., the plunger velocity at each time step will be determined from the balance between the hydraulic pressure force and metal pressure force on the plunger obtained from the results of previous step. This will give a better plunger deceleration profile.
- Investigating various deceleration profiles of the plunger velocity and their effects on the impact pressure.
- Applying the transient cavity pressure to determine the parting plane separation and its time duration using a full machine and die FEA model. Thus, relating the parameters of plunger shot profile and impact spike with size and persistence of the parting plane separation.
- Creating a regression model, or similar simple model, to provide a tool that can be used to approximately evaluate the effect of plunger deceleration on impact pressure and die separation

REFERENCES

Ahuett-Garza, H., Miller, R.A., “Die casting die deflections: Computer simulation of causes and effects”, Transactions of the 19th International Die Casting Congress & Exposition (1997)

Branden, J., Olmstead, P. and Kuhn, C.W., “Plunger Deceleration of a Die Cast Shot End – Applying Peter Olmstead SoftSHOT^{TM} Simulation Software”, NADCA Congress (2002)

Choudhury, A.K., “Study of the Effect of Die Casting Machine upon Die Deflections”, The Ohio State University, Master Thesis (1997)

Flow Science Inc., Flow-3D

Kesavan, V., “Development of Computer Simulation Models for Die Casting Studies”, The Ohio State University, Master Thesis (1999)

Mickowski, J.R. and Teufert, C.E., “The Control of Impact Pressure in the High Pressure Die Casting Process”, Transactions of the 17th NADCA Congress & Exposition (1993)

Ragab, A., “Sensitivity Analysis of Casting Distortion and Residual Stress Prediction through Simulation Modeling and Experimental Verification”, The Ohio State University, PhD dissertation (2003)

Savage, G., Gershenzon, M. and Rogers, K.J., “The Role of Pressure in High Pressure Die Casting”, NADCA Congress (2001)

Tong, K.K.S., Hu, B.H., Niu, X.P. and Pinwill, I., “Cavity Pressure Measurements and Process Monitoring for a magnesium Die Casting of a Thin-Wall Hand-Phone Component to Improve Quality”, Journal of Materials Processing Technology, Vol. 127, pp. 238-241 (2002)

Venkatasamy, V., Brevick, J., Mobley, C. and Pribyl, G., “Die Cavity Sensors for Monitoring Die Casting Processes”, Transactions of the 19th International Die Casting Congress & Exposition (1997)

Appendix D: Modeling Dynamic Pressure in High Pressure Die Casting Using Lumped Parameter Models

2011 DIE CASTING CONGRESS & TABLETOP NADCA

This paper is subject to revision. Statements and opinions advanced in this paper or during presentation are the author's and are his/her responsibility, not the Association's. The paper has been edited by NADCA for uniform styling and format. For permission to publish this paper in full or in part, contact NADCA, 241 Holbrook, Wheeling, Illinois, 60090, and the author.

Modeling Dynamic Pressure in High Pressure Die Casting Using Lumped Parameter Models

H. Xue, K. Kabiri-Bamoradian, R. A. Miller, Y. Wang
The Ohio State University, Columbus, OH, USA

ABSTRACT

The large dynamic pressure spike at the end of injection in high pressure die casting, due to the rapid deceleration of the liquid metal, plunger, and hydraulic oil, plays a key role in die flashing. The fact that the cavity pressure and plunger speed are tightly coupled makes prediction of transient cavity pressure extremely difficult using current fluid flow simulation approaches that use a specified velocity profile as an input. The purpose of this research is to develop a simple model that can predict transient cavity pressure, gain insight about the influence of the shot profiles on the cavity pressure spike, and also can be used as an input for die distortion modeling. A multi-degree of freedom lumped-parameter model that represents the injection system and die casting machine was developed and implemented in MatLAB/Simulink[1]. This model addresses the coupling issues between plunger speed and pressure and predicts the transient cavity pressure during cavity filling in a fast and computationally efficient way. Different shot plunger deceleration profiles were studied and useful insights into the die responses were achieved. The factors included in the study are deceleration start time, plunger position when velocity is zero, plunger position when controller switches to pressure control, and hydraulic pressure rise time to full accumulator pressure. The maximum cavity pressure was predicted with the aid of the above lumped-parameter model. A design of experiments study was performed to evaluate the effect of these factors on the cavity pressure spike. The results show quantitatively that a decelerated plunger shot profile can dramatically decrease the impact pressure spike.

INTRODUCTION

At the end of injection in the high-pressure die-casting process, a large spike in the dynamic metal pressure will be experienced. During the injection, the liquid metal is rapidly pushed into the die cavity by a plunger typically at more than 100 inches/sec. The molten metal, the plunger, plunger rod, hydraulic piston, and some amount of hydraulic oil are all in motion at a high speed during the injection and rapidly decelerate and stop at the end of filling. The sudden stop of all these moving masses can create a very high dynamic pressure in the die cavity – impact pressure. The cavity pressure is transient, decays relatively quickly, and the die will respond to this dynamic pressure. The extreme high pressure peak developed at the end of filling, combined with the dynamic responses of the die casting machine and dies, causes the die components to deflect and may result in die flashing and accordingly increased cleaning cost and cast part dimensional instability.

The role of dynamic cavity pressure was well studied by Savage, G. et al. Its influence on casting quality shows that high impact pressure generally does not improve the casting quality, whereas intensification pressure does. The high impact pressure, or high plunger velocity, makes flashing more likely [Savage, G. *et al*, 2001]. J. R. Mickowski et al stated that the impact pressure should be controlled such that the plunger velocity and final pressure are retained. He also introduced a real-time closed-loop shot control system to control the impact pressure. In this system, a rapid response valve was added to the hydraulic cylinder so that an extremely rapid deceleration in a small final fraction of the plunger stroke can be applied, by which overall cavity filling time at least as short as in the un-controlled conventional process was maintained [Mickowski, J.R. *et al*, 1993]. Peter Olmsted developed a very innovative idea that uses the overflows to cushion the impact pressure. By adjusting the overflow volume and the gate size to overflows, the impact pressure can be controlled in a very robust way [Branden, J. *et al*, 2002]. During the plunger deceleration phase, the cavity pressure and plunger movement are highly

[1] MATLAB/Simulink is a registered trademark of MathWorks.

coupled due to the fact that the interaction between the hydraulic pressure and metal pressure forces the plunger into oscillation, which decays to a full stop. Accurate pressure results especially the impact pressure spike at the end of filling cannot be obtained using an uncoupled simulation [Xue, H. *et al*, 2005].

Cavity pressure is very difficult to be accurately measured due to the complexity of die casting process and the very fast speed of response. Computer simulation of the injection process thus becomes a logical choice to predict the transient cavity pressure. However, in most fluid flow modeling, the pressure is computed from a prescribed plunger velocity, which is an uncoupled approach (velocity is independent of pressure). The general purpose CFD software FLOW-3D[2] has the capability of modeling fully coupled motion of solid bodies in fluids by applying FAVOR^{TM} technique to general moving objects (GMO), which can be used to simulate die cavity filling by modeling the plunger and allows for position dependent hydraulic driving force [Starobin A. *et al*, 2008]. However, numerical fluid volume loses that occur during this computation cause trouble in predicting the pressure. The molten metal is always assumed as either incompressible or slightly compressible fluid; therefore the pressure impact is very sensitive to the volume change of molten metal due to its very large bulk modulus. Pressure prediction may be meaningless if the numerical volume of molten metal changes obviously between time steps.

One goal of this research is to create a simple model that not only addresses the coupling effects between the plunger speed and metal pressure during the plunger deceleration, but also captures the cavity pressure efficiently without requiring the complexity of a 3-D simulation model. For this purpose, a multi-degree of freedom lumped-parameter model that integrates the injection system and components of die casting machine was developed and implemented in MatLAB's Simulink. The lumped-parameter model has many fewer degrees of freedom than a 3-D simulation model; therefore, insights can be achieved in a fast and inexpensive fashion but with a loss of accuracy and detail. The other goal of this research is to evaluate the effects of plunger deceleration profiles on cavity pressure spike. With the aid of a shot monitoring system, the above lumped-parameter model can serve as a convenient tool to accomplish this task.

MODELING APPROACH

The plunger deceleration stage is the focus of this research. Fig. 1 shows a typical injection system used in die casting. This injection system can be simplified as a single degree of freedom lumped-parameter model, as shown in Fig. 2.

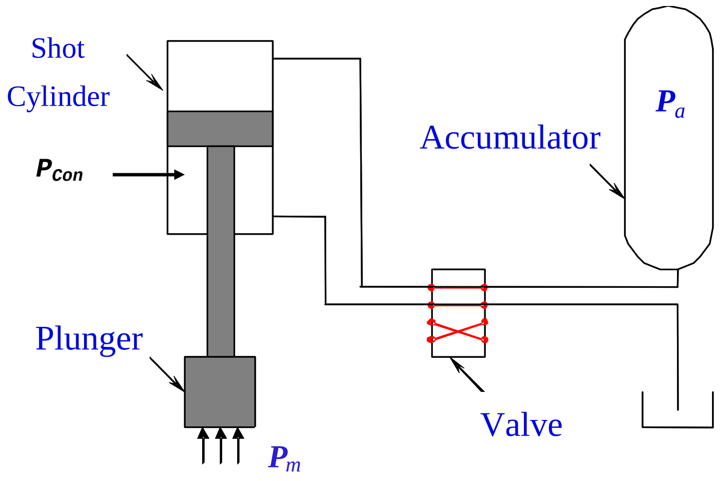


Fig. 1. Schematic of an Injection System

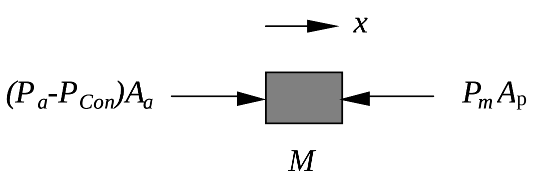


Fig. 2. A Lumped-Parameter Model Representing the Injection System Only

[2] FLOW-3D is a registered trademark of Flow Science Inc.

During the slow shot, acceleration, and fast shot phases of molten metal injection, the valve controls the speed of the plunger. The valve throttles the flow of hydraulic fluid out of the shot cylinder that in turn controls the pressure difference across the shot piston. This is captured in the lumped-parameter model by (P_a – P_{Con}). When the plunger reaches some specified position at the end of cavity filling, the output side of the valve is opened to the reservoir so that the full accumulator pressure is applied to the piston. The simplified equations of plunger motion are as follows:

$$M\ddot{x}(t) + C\dot{x}(t) = (P_a(t) - P_{Con}(t))A_H - P_m(t)A_p \qquad (1)$$

where the pressures are defined as in Fig. 1. M is the total moving masses including the moving parts of injection system, some hydraulic oil, and molten metal. A_H and A_p are cross-sectional areas of hydraulic piston and plunger tip, respectively.

The machine control system will throttle the pressure on the backside of the shot piston, P_{Con}, so that the programmed velocity profile is tracked as closely as possible. Mathematically, the resulting control is of proportional plus integral form

$$P_{Con}(t) = K_1(\upsilon_c(t) - \dot{x}(t)) + K_2\int_0^t(\upsilon_c(\tau) - \dot{x}(\tau))d\tau \qquad (2)$$

State variable representation of the system is useful for the analysis and is easily constructed. Using the plunger position and velocity as state variables, i.e.,

$$z_1(t) = x(t)$$
$$z_2(t) = \dot{x}(t) \qquad (3)$$

The control equation (2) is also simplified by using state variables. Defining a 3rd state variable to account for the integral in the control provides

$$\dot{z}_3(t) = \upsilon_c(t) - \dot{x}(t) = \upsilon_c(t) - z_2(t) \qquad (4)$$

Combining equations above gives the closed-loop equations of motion.

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{C}{M} + \frac{A_H K_1}{M} & -\frac{A_H K_2}{M} \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{A_H}{M} & -\frac{A_p}{M} & -\frac{A_H K_1}{M} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_a \\ P_m \\ \upsilon_c \end{bmatrix} \qquad (5)$$

Equation (5) applies as long as

$$z_1(t) < x_{threshold}$$

In other words, the equations apply as long as the cavity is not full and the plunger position is before the threshold at which the control pressure is dropped to zero. The cavity full point corresponds to the point at which the volume of molten metal in the shot equals to the sum of the cavity volume, the runner volume and the biscuit volume. The impact spike will begin at this point or a little earlier.

When the plunger threshold position is reached at the end of filling, the velocity control is turned off and the control pressure is developed to zero so that the accumulator pressure is applied in full to the shot position. The equations of motion then reduce to

$$\begin{bmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{C}{M} \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ \frac{A_H}{M} & -\frac{A_p}{M} \end{bmatrix} \begin{bmatrix} P_a(t) \\ P_m(t) \end{bmatrix} \qquad (6)$$

Equations (5) and (6) define the needed equations of motion for coupling with fluid flow simulation. The metal pressure P_m is computed throughout the stroke using the CFD software, for example, Flow-3D. With this as an input to the equations, the lumped model is used to determine the plunger velocity required by Flow-3D. A subroutine is needed to fulfill this function if using Flow-3D.

For a lumped approximation without a cavity filling simulation, we need a way to approximate the gate resistance and the effect of the full cavity on the pressure. These can be roughly accomplished by using the bulk modulus to account for the cavity full condition and using an approximate Bernolli relationship to capture the gate resistance. Considering the gate resistance first, applying Bernolli's principle to both sides of the gate leads to equation (7). P_{cav}(t) denotes the pressure inside the cavity and the coefficient c_d is an empirical discharge coefficient that accounts for energy loses through the gate and ρ is the molten metal density.

$$P_m(t) - P_{cav}(t) = \frac{\rho}{2}\left[\left(\frac{A_p}{c_d A_g}\right)^2 - 1\right] z_2(t)^2 \tag{7}$$

Equation (7) provides the pressure differential across the gate as a function of the plunger velocity, plunger cross-sectional area and the gate area. This equation is valid only for streamline flow and can be viewed only as an approximation for a transient situation such as this, but it is still a useful approximation. Under this assumption, equation (5) becomes

$$\begin{bmatrix} \dot{z}_1 \\ \dot{z}_2 \\ \dot{z}_3 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{C}{M} + \frac{A_H K_1}{M} & -\frac{A_H K_2}{M} \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{A_H}{M} & -\frac{A_p}{M} & -\frac{A_H K_1}{M} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_a \\ P_{cav} \\ \upsilon_c \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \frac{\rho}{2}\frac{A_p}{M}\left[\left(\frac{A_p}{c_d A_g}\right)^2 - 1\right] z_2(t)^2 \qquad (z_1(t) < x_{threshold}) \tag{8}$$

For purpose of approximation, we can assume that the cavity pressure is almost zero until the cavity is full or nearly full. In order to track these conditions we need to compute the volume of metal, V_f, in the cavity.

$$\frac{dV_f(t)}{dt} = \begin{cases} 0, & if \quad z_1(t) < x_{gate} \\ z_2(t)A_p, & if \quad x_{gate} \le z_1(t) < x_{full} \\ -z_2(t)A_p, & if \quad x_{full} \le z_1(t) \end{cases} \tag{9}$$

The different forms of equation (9) reflect the fact that the cavity does not start until the metal reaches the gate and once the cavity fills, the volume of metal must decrease as the plunger tries to compress the metal. At this point the pressure will increase drastically. We account for the pressure increase via the bulk modulus of the molten metal.

$$dP_{cav} = -E_B \frac{dV}{V} \tag{10}$$

The modulus E_B is a large number and the cavity pressure will increase rapidly if the volume decreases, i.e., the system tries to compress the molten metal. Rewriting equation (10) in terms of the volume filled, and combining with equation (9) to provide the needed equations in terms of velocity.

$$\frac{dP_{cav}(t)}{dt} = \begin{cases} 0, & if \quad z_1(t) < x_{full} \\ \frac{E_B A_p}{V_f(t)} z_2(t), & if \quad z_1(t) \ge x_{full} \end{cases} \tag{11}$$

Equation (11) is solved with initial condition $P_{cav}(0)=0$ which means that the cavity pressure will be zero until the cavity full position is reached.

Once the plunger threshold position is reached at the end of filling, i.e., $z_1(t) \ge x_{threshold}$, the velocity control system is turned off and the control switches to pressure control. The corresponding equations under a ramp-down pressure control is developed as follows

$$\begin{bmatrix} \dot{z}_1(t) \\ \dot{z}_2(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{C}{M} \end{bmatrix} \begin{bmatrix} z_1(t) \\ z_2(t) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 \\ \frac{A_H}{M} & -\frac{A_p}{M} & -\frac{A_H}{M} \end{bmatrix} \begin{bmatrix} P_a \\ P_{cav} \\ P_{con} \end{bmatrix} - \begin{bmatrix} 0 \\ 1 \end{bmatrix} \frac{\rho}{2}\frac{A_p}{M}\left[\left(\frac{A_p}{c_d A_g}\right)^2 - 1\right] z_2(t)^2 \tag{12}$$

Where P_{con} is called controlled pressure and has the following definition

$$P_{con} = \begin{cases} P_{switch}, & (t = t_{switch}) \\ P_{switch}\left(1 + \frac{t_{switch}}{\Delta} - \frac{t}{\Delta}\right), & (t_{switch} < t < t_{switch} + \Delta) \\ 0, & (t \ge t_{switch} + \Delta) \end{cases} \tag{13}$$

P_{switch} is the controlled pressure when plunger reaches the threshold position and Δ is the duration that P_{con} changes from P_{switch} to zero. Equations (9) and (11) still apply.

The complete solution for the lumped approximation can now be summarized based on equations (8), (9), (11), (12) and (13).

In order to study the effects of other die machine components, especially the accumulator system, on the impact cavity pressure, a more complicated multi-degree-of-freedom lumped mass-spring-dashpot model was proposed based on the single-degree-of-freedom model mentioned above. Since the accumulator system and die casting machine are taken into consideration in this model, more accurate prediction of impact pressure spike and insights are expected in a fast and inexpensive way. This lumped-parameter model was illustrated in Fig. 3, which is a 3-degree-of-freedom model (3-DOF).

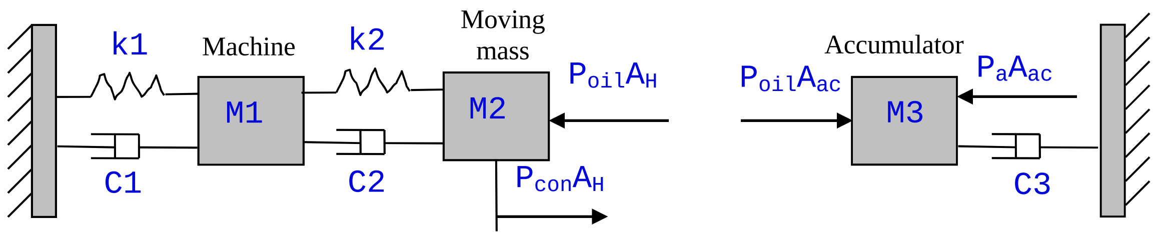


Fig. 3. A 3-DOF Lumped-parameter Model

The governing equations for the above system are

$$\begin{aligned} M_1\ddot{x}_1 &= -k_1x_1 - C_1\dot{x}_1 - k_2(x_1 - x_2) - C_2(\dot{x}_1 - \dot{x}_2) \\ M_2\ddot{x}_2 &= (P_{oil} - P_{con})A_H + k_2(x_1 - x_2) + C_2(\dot{x}_1 - \dot{x}_2) \\ M_3\ddot{x}_3 &= P_aA_{ac} - P_{oil}A_{ac} - C_3\dot{x}_3 \end{aligned} \tag{14}$$

Where

M_1, k_1, and C_1 – total mass, stiffness, and damping of Die and machine components
M_2 – total mass of moving components (liquid metal, hydraulic oil, and piston and plunger)
k_2 – equivalent stiffness due to the Bulk Modulus of liquid metal
C_2 – damping of injection system
M_3 – total mass of accumulator piston and some hydraulic oil
C_3 – damping of accumulator system
P_{oil}, P_a and P_{con} – pressure of hydraulic oil, accumulator pressure, and controlled pressure against the hydraulic piston
A_H, A_{ac} – cross-sectional areas of hydraulic piston and accumulator piston
k_2 can be determined by bulk modulus E_B, plunger tip area A_P, and cavity volume V_{cavity} as below

$$k_2 = \frac{E_BA_p^2}{V_{cavity}} \tag{15}$$

Any small change of the oil volume between the hydraulic piston and accumulator piston can cause dramatic oil pressure change since the oil is almost incompressible. Therefore, the connection between M_2 and M_3 is established and the following equation holds:

$$\frac{dP_{oil}}{dt} = \frac{B_{oil}}{V_{oil}}(\dot{x}_2A_H - \dot{x}_3A_{ac}) \tag{16}$$

where B_{oil} is the oil bulk modulus. Equations (14), (15) and (16) then can be used to solve the 3-DOF system with four unknowns $P_{oil}, \dot{x}_1, \dot{x}_2, \dot{x}_3$.

If the amount of hydraulic oil displaced by hydraulic piston and by accumulator piston is assumed to be the same during the injection, then x_2, x_3 are dependent and meet the relationship $A_Hx_2 = A_{ac}x_3$. Let $\beta = A_H / A_{ac}$, then we have

$$x_3 = \beta x_2,\ \dot{x}_3 = \beta\dot{x}_2,\ \ddot{x}_3 = \beta\ddot{x}_2 \tag{17}$$

The time dependent accumulator pressure can be evaluated by nominal accumulator pressure P_0, piston area ratio β, and initial height of nitrogen α:

$$P_a = \frac{P_0}{1 + \alpha\beta x_2} \tag{18}$$

Thus the 3-DOF model can be simplified to a 2-DOF model, as shown in Fig. 4.

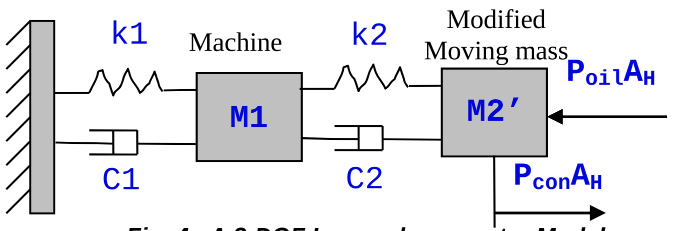


Fig. 4. A 2-DOF Lumped-parameter Model

Substitutes (17), (18), (15) into (14), then governing equations can be rearranged as

$$\begin{aligned} &M_1\ddot{x}_1 = -k_1x_1 - c_1\dot{x}_1 - k_2(x_1 - x_2) - c_2(\dot{x}_1 - \dot{x}_2) \\ &(M_2 + \beta^2 M_3)\ddot{x}_2 = P_a\beta A_{ac} - c_3\beta^2\dot{x}_2 - P_{con}A_H + c_2(\dot{x}_1 - \dot{x}_2) + k_2(x_1 - x_2) \end{aligned} \tag{19}$$

Above equations represent the 2-DOF model.

Matlab/Simulink was used to solve the above Ordinary Differential Equations (ODEs) that represent the injection process. A good estimate of the system masses can be obtained by computing the mass of the plunger tip and rod, the shot piston and rod, the mass of die machine components, and the mass of the moving hydraulic fluid. The molten metal mass can be adequately estimated by computing the shot volume during the cavity filling. Dry shot data were used to estimate the system and control parameters for velocity control. The damping was determined based on the time that the die and machine take to come to rest. Then the control coefficients K's were calculated according to the decay ratio and frequency from the dry shot plots. The model takes programmed plunger speed as the input and calculates slow shot, acceleration, and fast shot under the velocity control. It switches to pressure control when the plunger reaches some certain position, for example, 99% full of cavity. A ramp-down pressure profile, which was determined by the pressure at switch point and the final static pressure, was applied and the system responses for the rest of plunger stroke and after-cavity-full were continually calculated.

The procedure is the same for coupling with numerical simulation, except that there is no need to use the Bernoulli equation and instead the simulation is used to compute the metal pressure directly.

RESULTS AND DISCUSSION

Three lumped-parameter models with one, two, and three degrees of freedom respectively were implemented using MATLAB's Simulink. The tests and comparisons are based on these models.

COMPARISON OF 1-, 2-, 3-DOF LUMPED-PARAMETER MODELS

A plunger shot profile, defined by constant fast shot speed, no deceleration at the end of filling, and a pressure profile that full accumulator pressure applies linearly within 50 milliseconds after plunger reaches the threshold position, was tested with 1-, 2-, and 3-DOF models. Results are shown in Fig. 5~8.

The results show that the maximum cavity pressure is 2.0e8 Pa for single-DOF model; while it is 1.71e8 Pa for the 2-DOF model and 1.75e8 Pa for 3-DOF model. This makes sense in that the single-DOF model is "stiffer" than the multi-DOF model due to the addition of the machine and accumulator systems in multi-DOF models. Also because of the existence of the 2nd and/or 3rd masses, the decay of response is no longer as smooth as the decay of the oscillation in the single DOF model. Instead superimposed waveforms due to the additional response modes are observed, as shown in Fig. 8. In addition, the cavity pressure is dissipated more rapidly for multi-DOF models because multi-DOF model allows the die to move and deflect, to some degree.

Comparing Fig. 6 and Fig. 7, it can be seen that there is not much difference in cavity pressure prediction between the 2-DOF and 3-DOF models. This is true as long as the accumulator damping is not too high. This issue is discussed in the next section.

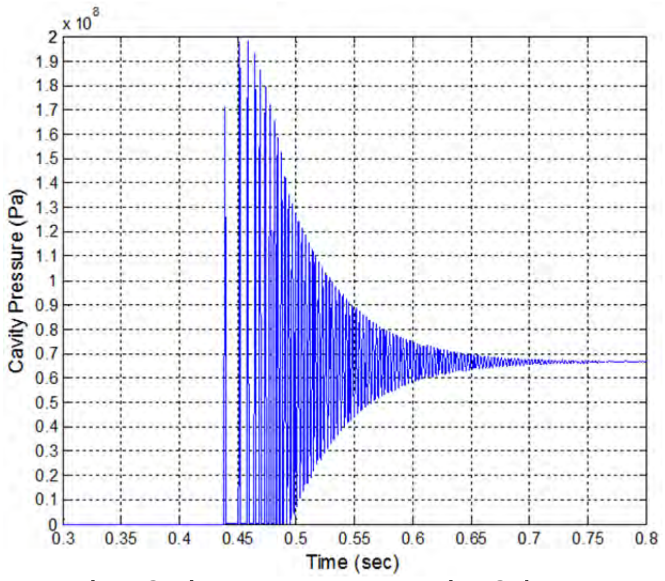


Fig. 5 Cavity pressure response (1-DOF)

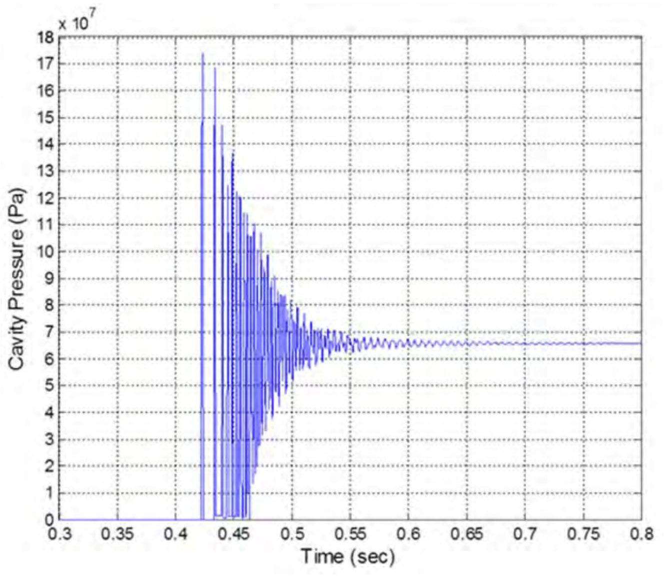


Fig. 6 Cavity pressure response (3-DOF)

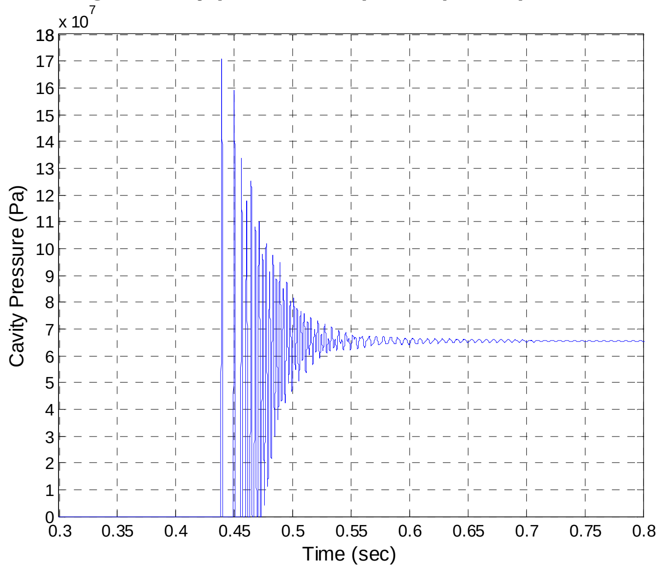


Fig. 7 Cavity pressure response (2-DOF)

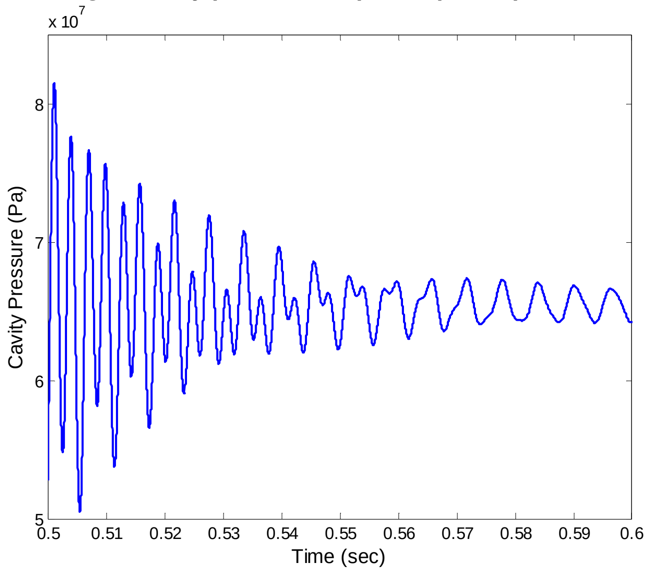


Fig. 8 Superimposed Cavity pressure response

INVESTIGATION OF ACCUMULATOR'S DAMPING EFFECT

One parameter that is difficult to estimate from response data of from first principles is the accumulator damping. Therefore, a sensitivity study was used to better understand the relative importance of this parameter. In previous cases, the damping from the accumulator for the 2- and 3-DOF models is set at 40,000 N.s/m, which meets the critical damping requirement. A typical accumulator pressure response is shown in Fig. 9. During the slow shot, the accumulator pressure drops very slowly since the accumulator piston moves slowly; when fast shot begins, the accumulator pressure decreases rapidly because of the relatively fast speed of the piston; finally accumulator pressure stays at a constant, about 98.6% of full accumulator pressure due to the fact that the cavity is full and the plunger cannot move.

The corresponding responses of cavity and accumulator pressure are shown in Fig. 10 ~13 for a high damping case (damping equals 2.0e7 N.s/m each case). It was observed that under high damping the 2-DOF model developed an unrealistic pressure profile for both cavity pressure and accumulator pressure. With high damping, the accumulator piston's movement is very limited. This causes accumulator pressure to drop very slowly and thus the hydraulic pressure on hydraulic piston rises more slowly than needed. Therefore, it takes much longer time for cavity and accumulator pressures to reach their static pressures, as indicated in Fig. 11 and 13. However, for 2-DOF model, since two pistons' movements meet fixed relationship, it created an artificially high pressure inside the cavity as shown in Fig. 10.

.

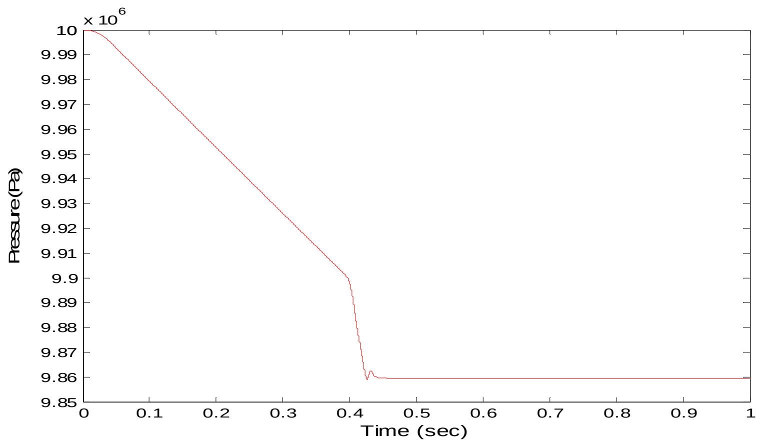


Fig. 9. A Typical Accumulator Pressure Profile

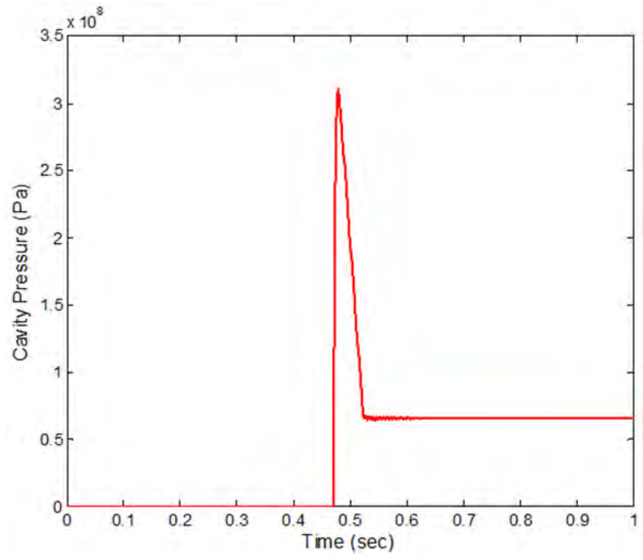


Fig. 10. Cavity Pressure (2-DOF)

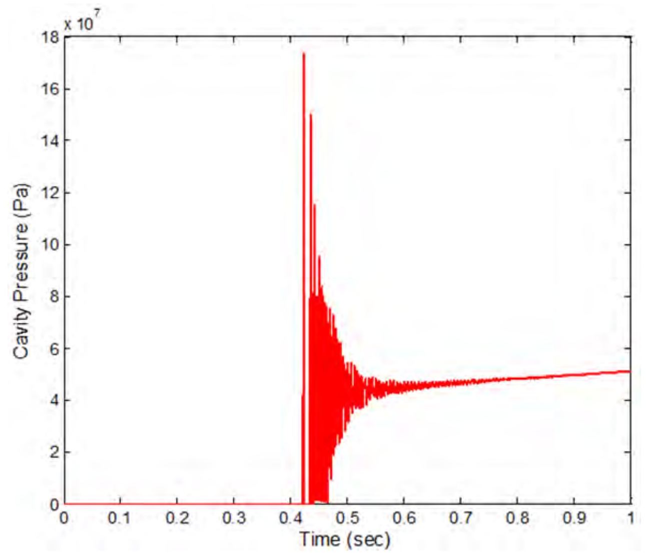


Fig. 11. Cavity Pressure (3-DOF)

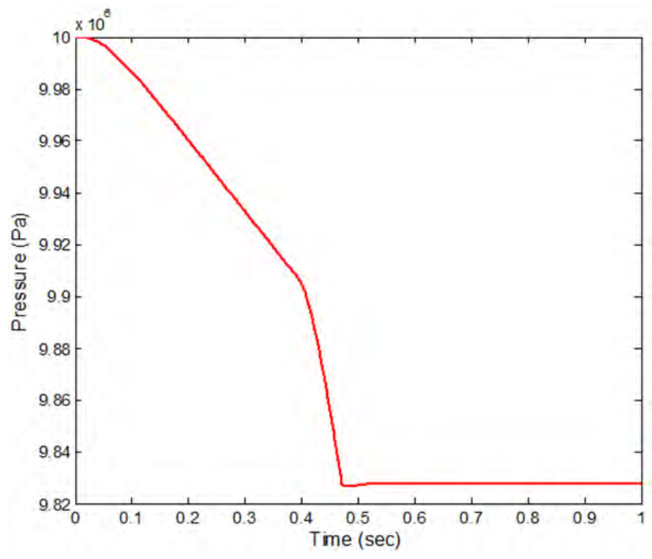


Fig. 12. Accumulator Pressure (2-DOF)

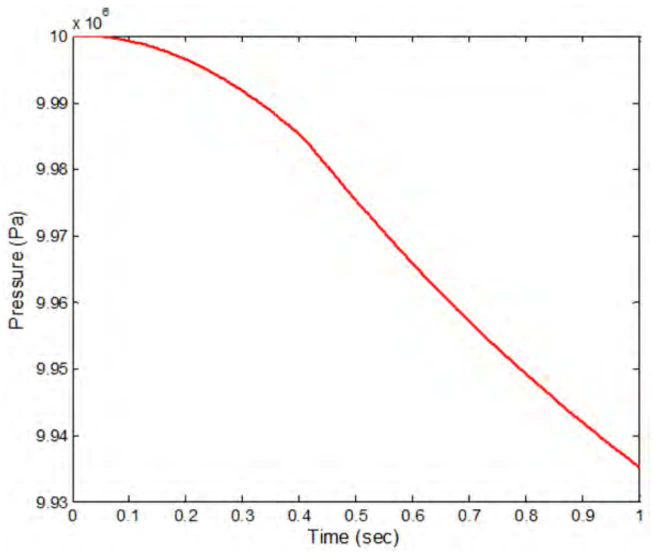


Fig. 13. Accumulator Pressure (3-DOF)

As pointed out previously, as long as the damping is low (not greater than critical), the accumulator damping has almost no influence on the dynamic cavity pressure response. In such cases the 2 and 3 DOF models have virtually identical responses.

SPECTRAL DENSITY ANALYSIS

Spectral density analysis under normal damping (40000 N.s/m) was performed in order to examin the frequency contents and identify periodicities. It describes how the power of a signal is distributed with frequency. The 3-DOF model was used to perform this analysis. Results for the hydraulic and cavity pressures are shown as in Fig. 14 and 15. Two dominant frequencies, 170Hz and 340Hz, were observed for both pressures. No obvious 3rd frequency was detected for the 3-DOF model based on given model parameters.

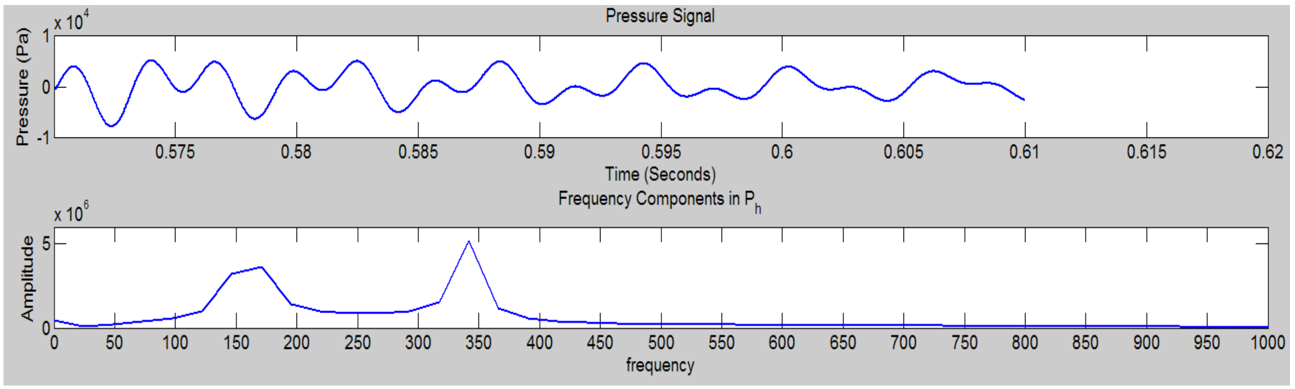


Fig. 14. Spectral Density Analysis for Hydraulic Pressure

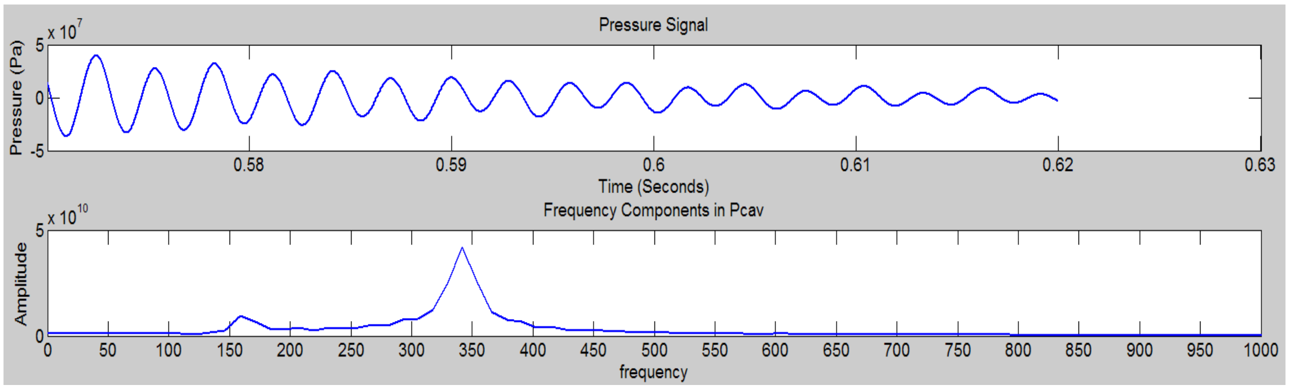


Fig. 15. Spectral Density Analysis for Cavity Pressure

PRESSURE PEAK AS A FUNCTION OF PLUNGER SHOT PROFILES

A decelerated plunger shot profile near the end of filling can decrease the impact pressure spike. But it is still unclear what deceleration profile should be used to minimize the cavity pressure spike; for instance, when to start the deceleration and how to decelerate. The plunger deceleration profiles and their effects on cavity pressure spike were explored using a set of computational experiments setup using a design of experiments.

Four process parameters that can control the plunger deceleration were investigated in this paper: deceleration start time, plunger zero velocity position, plunger position when velocity control switches to pressure control, and rising time to full accumulator pressure. The descriptions of the factors along with their levels are shown in Table 1. The response, maximum cavity pressure, was predicted with the aid of the 3-DOF lumped-parameter model. A design of experiment study was performed, in which both linear and quadratic regression models were created to evaluate the effect of these factors on the cavity pressure spike.

The definition of each factor and their relations are shown in Fig. 16. Fig. 17 shows the plunger deceleration profiles studied in this paper; dashed line is the spline shape deceleration profile and solid line is linear deceleration profile. Note that the spline profile has smoother transitions, but the start of significant deceleration is comes later than is the case for the linear profile and the rate of deceleration is generally higher.

Table 1 Factors used in Design of Experiments

Factor	Description	Low Level	Medium Level	High Level
A	Deceleration start time	0.41644 s	0.41788 s	0.41932 s
B	Plunger zero velocity position	26.432 cm	26.449 cm	26.462 cm
C	Plunger position where velocity control switches to pressure control	26.371 cm	26.401 cm	26.431 cm
D	Rising time of hydraulic pressure from zero to full accumulator pressure	0.01 s	0.03 s	0.05 s

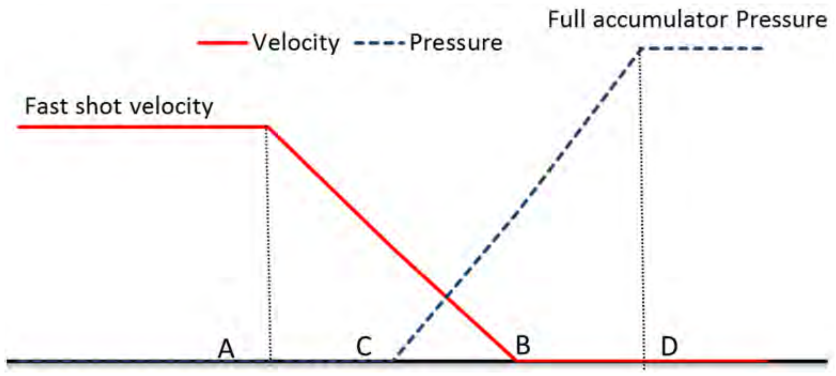


Fig. 16 Factor Definition Illustration

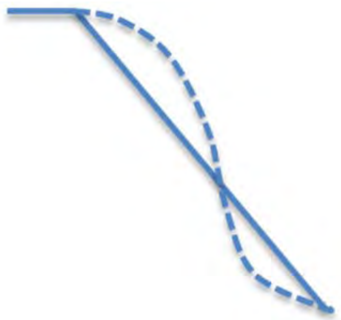


Fig. 17 Deceleration Profiles

A 27 run Box-Behnken response surface experimental design was chosen based on the four factors [Montgomery, D.C., Myers, R.H]. The experimental array in un-coded units is shown in Table 2.

The 3-DOF lumped-parameter model was used to predict the impact pressure peak. Both linear and spline plunger deceleration profiles were investigated using quadratic regression models.

<u>Linear deceleration profile</u>

The fit model was calculated using statistical software MINITAB and shown as below: [3]

$$P_m = 183057036 + 353307035A - 307500B + 9529747C + 10599696D$$
$$-3613750A^2 + 1854214B^2 + 1570000C^2 + 3320000D^2 - 10747895AB + 3955000AC$$
$$-662500AD - 2181534BC - 6024707BD + 3340000CD$$

Fig. 18 shows the interaction among these factors. It can be observed that interactions between factors A and B, factors B and D are significant; indicating the deceleration slope and rising time to full accumulator pressure are of importance for cavity pressure. In order to minimize the impact pressure, extra care must be taken when selecting levels for these interacting factors. In addition, the sign of the effect of each factor has to be taken into consideration. It can be observed that no squared terms are significant. By looking at both interaction plot and the effect of each interaction, we conclude that the minimum cavity impact pressure is achieved when all four factors are set to their lower levels. This can be verified from the contour interaction plots; the gradients in each plot points to the lower-left corner when the other factors are held at their lower levels, as shown in Fig. 19.

[3] The variables A, B, C, and D are the factors defined in Table 1.

Table 2. Response Surface Experimental Array

Run	A (s)	B (cm)	C (cm)	D (s)
1	0.41644	26.449	26.401	0.01
2	0.41644	26.449	26.371	0.03
3	0.41932	26.462	26.401	0.03
4	0.41788	26.462	26.401	0.01
5	0.41788	26.432	26.401	0.01
6	0.41644	26.462	26.401	0.03
7	0.41788	26.462	26.371	0.03
8	0.41788	26.432	26.431	0.03
9	0.41932	26.449	26.401	0.05
10	0.41788	26.449	26.371	0.05
11	0.41644	26.449	26.401	0.05
12	0.41932	26.432	26.401	0.03
13	0.41932	26.449	26.431	0.03
14	0.41932	26.449	26.371	0.03
15	0.41788	26.432	26.371	0.03
16	0.41788	26.449	26.431	0.01
17	0.41788	26.449	26.401	0.03
18	0.41644	26.449	26.431	0.03
19	0.41788	26.462	26.431	0.03
20	0.41932	26.449	26.401	0.01
21	0.41788	26.449	26.431	0.05
22	0.41788	26.432	26.401	0.05
23	0.41788	26.449	26.401	0.03
24	0.41788	26.449	26.401	0.03
25	0.41788	26.462	26.401	0.05
26	0.41788	26.449	26.371	0.01
27	0.41644	26.432	26.401	0.03

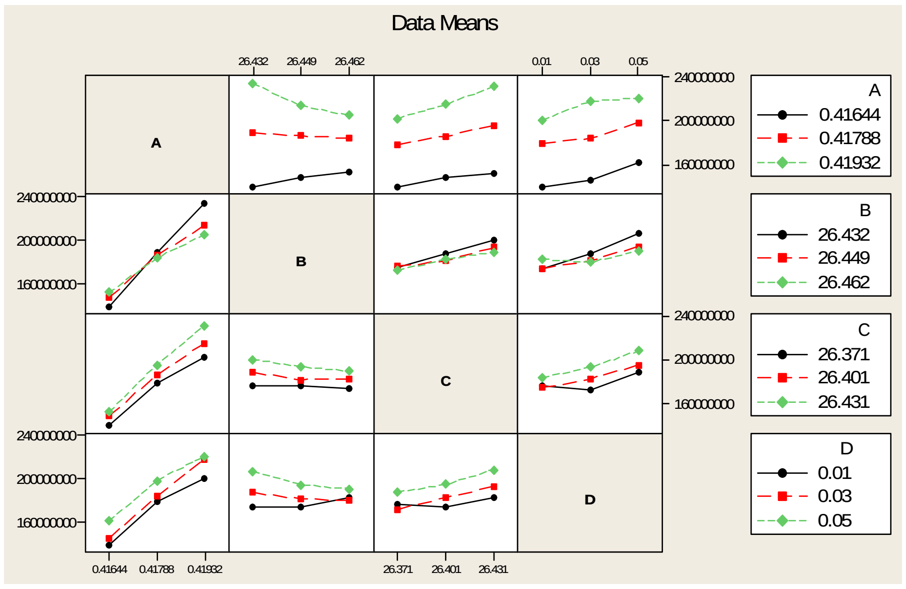


Fig. 18. Interaction Plot for Linear Deceleration

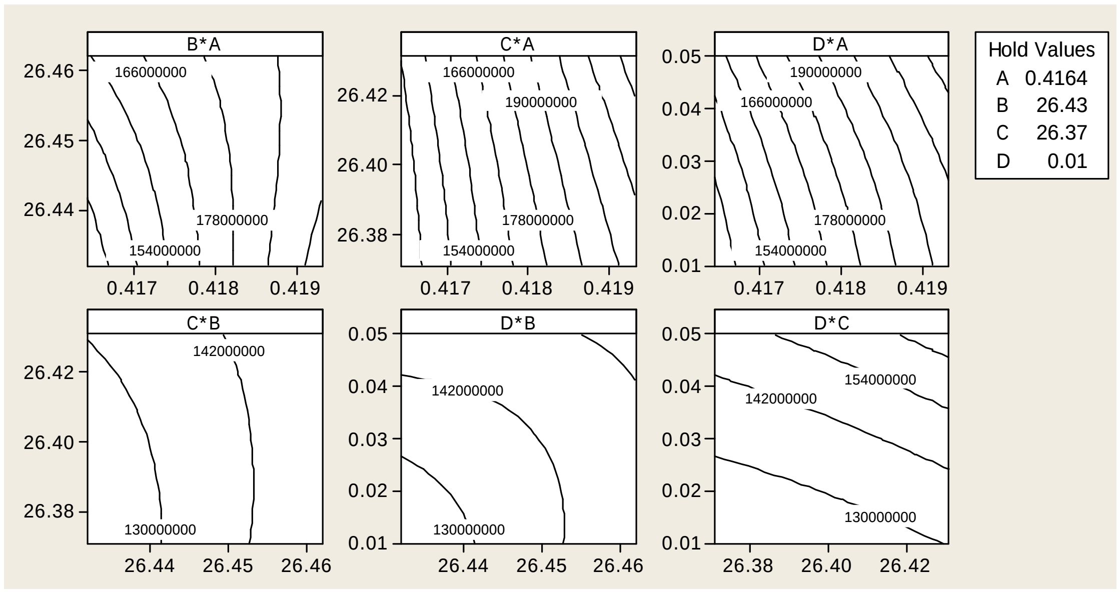


Fig. 19. Contour Plot for Linear Deceleration

<u>Spline deceleration profile</u>

The fit model was calculated using statistical software MINITAB and shown as below: [4]

$$P_m = 241280505 + 215598265A - 6761667B + 14960706C + 15876819D$$
$$-12347917A^2 - 15690922B^2 - 11851667C^2 - 15184167D^2 - 74231AB + 90000AC$$
$$-420000AD - 1348572BC - 10061091BD + 3472500CD$$

The interaction effects are shown in Fig. 20. It is observed that factors A, B and C have no interaction with other factors. Other terms are significant. In order to achieve minimum impact pressure, the contour plot, Fig. 21, suggests again that all factors take their lower levels.

[4] The variables A, B, C, and D are the factors defined in Table 1.

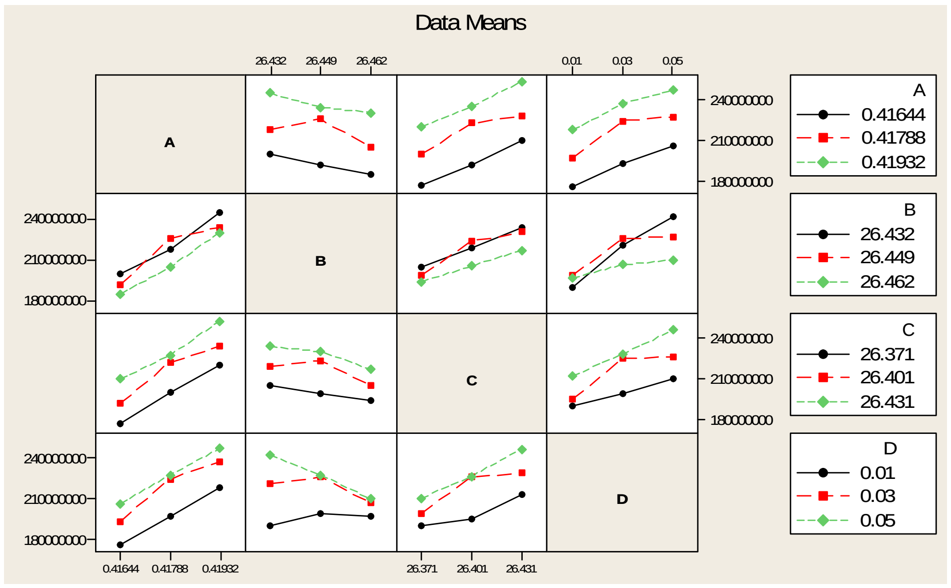


Fig. 20. Interaction Plot for Spline Deceleration

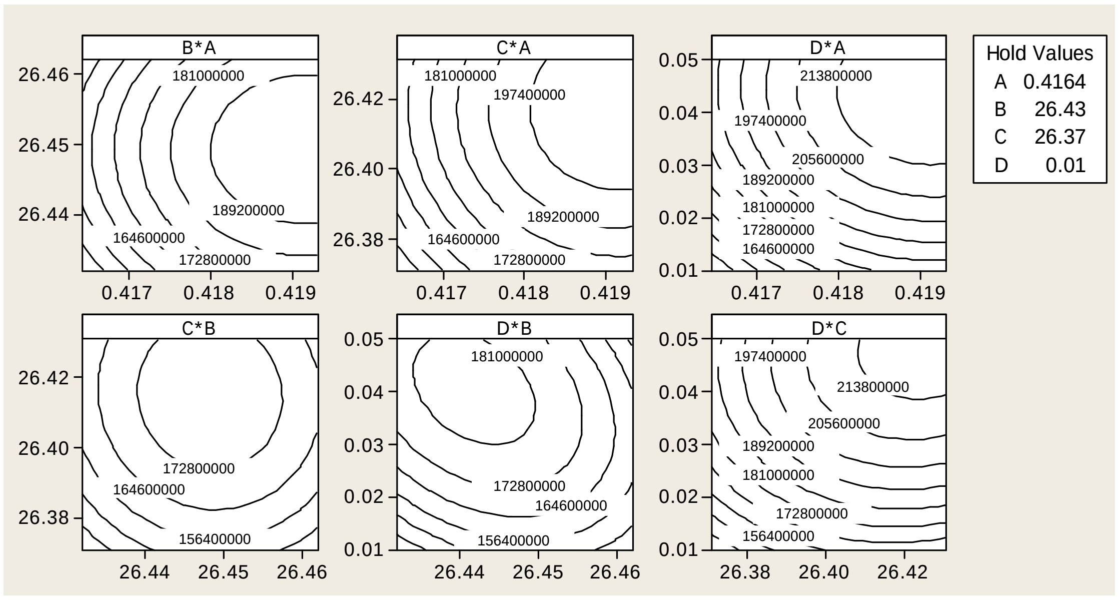
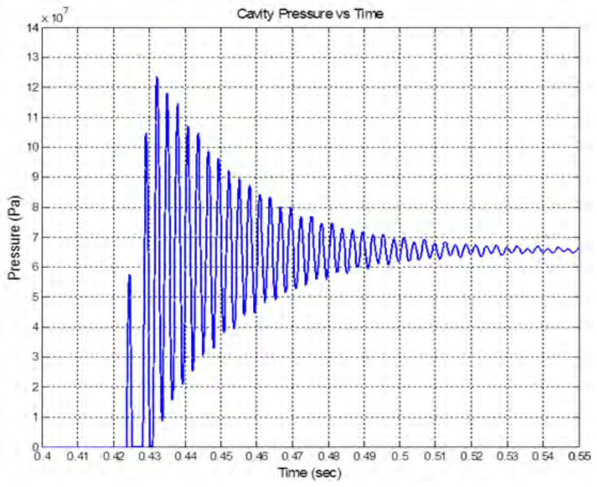


Fig. 21. Contour Plot for Spline Deceleration

The cavity pressure responses when using optimal levels are plotted in Fig. 22 and 23. With linear deceleration profile, the maximum pressure peak is about 1.23e8 Pa; while the maximum pressure is around 1.6e8 pa with the spline deceleration profile. Linear deceleration is more effective in reducing the impact pressure because velocity is higher for a slightly longer period of time with spline deceleration profile compared to linear deceleration. Without plunger pre-deceleration and ramp-up hydraulic pressure, the impact pressure would be as high as 2.60e8 Pa when using the same lumped-parameter model so both profiles significantly reduce the spike.



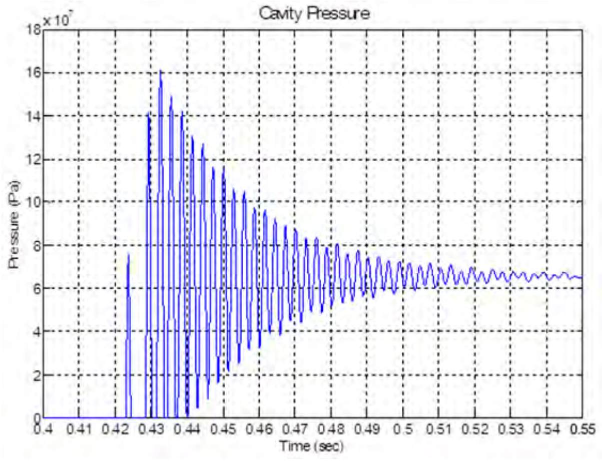


Fig. 22. Optimal Pressure (linear deceleration)

Fig. 23. Optimal Pressure (spline deceleration)

CONCLUSIONS AND FUTURE WORK

Multi-degrees of freedom lumped-parameter models that incorporate the injection system and die casting machine components have been established in this study. A frame work to develop strategies to evaluate dynamic effects without requiring the complexity of a full dynamic finite element model also have been initiated in this study; the lumped-parameter model of the dynamic system and regression models of the relationship between profile parameters and cavity pressure are good choices to fulfill this goal.

Die design, machine design, accumulator design, and shot profile all play a key roles in reducing the cavity pressure spike developed at the end of injection. The lumped-parameter model provides a quick and inexpensive tool to estimate/evaluate the different design parameters. The study shows that smaller pressure impact is predicted with multi-degree of freedoms model, which is more realistic because the multi-DOF system is not as rigid as the single-DOF system. The study also indicates that the 2-DOF and 3-DOF lumped-parameter models are very similar as long as the accumulator damping is not abnormally high.

Two regression models for the linear and spline deceleration profiles were used to study the effects of deceleration parameters on impact pressure. The regression models consistently show that levels of four investigated factors should be set at their low levels in order to minimize cavity impact pressure spike; i.e. the plunger should decelerate as early as possible with a steep slope of deceleration profile that reaches zero velocity as soon as possible and is consistent with the requirement that fill time is met; the pressure control should also start early and reach full accumulator quickly.

Ongoing activities in this research include:

- Improving regression model, or similar simple model, to provide a tool that can be used to approximately evaluate the effect of plunger deceleration on impact pressure and die separation.
- Modifying the fluid flow modeling by integrating with lumped-parameter model. For example, customize Flow-3D with subroutines to address the coupling between back-pressure and plunger velocity by using lumped approximation.
- Analyzing dynamic parting plane separation by replacing hydrostatic pressure with transient cavity pressure obtained from pressure model

REFERENCES

1. Ahuett-Garza, H., Miller, R.A., “Die casting die deflections: Computer simulation of causes and effects”, Transactions of the 19th International Die Casting Congress & Exposition (1997)
2. Branden, J., Olmstead, P. and Kuhn, C.W., “Plunger Deceleration of a Die Cast Shot End – Applying Peter Olmstead SoftSHOT^{TM} Simulation Software”, NADCA Congress (2002)
3. Choudhury, A.K., “Study of the Effect of Die Casting Machine upon Die Deflections”, The Ohio State University, Master Thesis (1997)

4. Montgomery, D.C., Myers, R.H. "Response Surface Methodology: Process and Product Optimization Using Designed Experiments" , ISBN 0-471-41255-4 (2002)
5. Kesavan, V., "Development of Computer Simulation Models for Die Casting Studies", The Ohio State University, Master Thesis (1999)
6. Mickowski, J.R. and Teufert, C.E., "The Control of Impact Pressure in the High Pressure Die Casting Process", Transactions of the 17th NADCA Congress & Exposition (1993)
7. Ragab, A., "Sensitivity Analysis of Casting Distortion and Residual Stress Prediction through Simulation Modeling and Experimental Verification", The Ohio State University, PhD dissertation (2003)
8. Savage, G., Gershenzon, M. and Rogers, K.J., "The Role of Pressure in High Pressure Die Casting", NADCA Congress (2001)
9. Venkatasamy, V., Brevick, J., Mobley, C. and Pribyl, G., "Die Cavity Sensors for Monitoring Die Casting Processes", Transactions of the 19th International Die Casting Congress & Exposition (1997)
10. Xue, H., Kabiri-Bamoradian, K., Miller, R. "Modeling Dynamic Cavity Pressure and Impact Spike in Die Casting", CastExpo/NADCA (2005)
11. Reikher, A., Gerber, H., and Starobin, A., "Multi-Stage Plunger Deceleration System", CastExpo/NADCA (2008)

Appendix E: Design Support for Tooling Optimization

Design Support for Tooling Optimization

R. A. Miller

D. Wang

The Ohio State University, Columbus, OH, USA

Copyright 2005 American Foundry Society

ABSTRACT

Several years ago qualitative reasoning methods were utilized to develop techniques to visualize the approximate fill pattern of high pressure die castings. Related methods were used to evaluate the castability of a part based on rapidly detecting the range of variation in wall thickness, ie, finding thick and thin sections. These methods were implemented in the *CastView* program distributed by NADCA. Recent research has extended the methods used for fill pattern visualization to include slower fill processes such as gravity, permanent mold, and squeeze casting. This enables a quick and dirty check of the fill pattern for the slower fill processes.

In addition, methods have been developed to compute the quasi-steady state temperature distribution in the die averaged over the casting cycle die casting without requiring the simulation of numerous casting cycles in order to reach quasi-steady state. This saves considerable computation time since the result is obtained in one step instead of the 20 or more if the transient is simulated. The average temperature distribution at quasi-steady state provides the information needed for cycle design and for cooling line placement in the die and, since the computation time needed to obtain the temperature distribution using the equilibrium technique is very fast, computational improvement or even optimization of cooling line placement is feasible.

The focus of this paper is on search-based optimization techniques that utilize the very short computation times achievable for both fill pattern and temperature distribution using the techniques mentioned above to guide the selection of gate locations and cooling line placement.

Introduction

During the die casting process, liquid or semi-solid metal is injected at high speed to fill a complex die cavity, and solidifies rapidly under high pressure. It is a complicated physical-chemical process. An incorrect process design often causes die casting defects, such as porosity and deformation, and also shortens the die life. To realize appropriate thermal balance and mold filling thus to ensure quality die casting products and prolong the die life, are two important issues for the die casting process design (Allchin, 1990).

The thermal characteristics of the die and casting are always fundamental considerations for designers. Early in design, the typical questions about thermal characteristics include: 1) What is the spatial temperature distribution of the die and part? 2) Where are the hot spots? 3) What is the effect of cooling lines and spray? 4) What is the ejection temperature of the part? Numerical simulation of the transient temperature is the typical way to obtain the answer. The typical procedure for numerical simulation is to build CAD models for part and dies – input thermal property data and cycle parameters – run simulation – view analysis results. The analysis results include the temporal and spatial temperature pattern of the part and dies. Thus the user can read temperature at any location and at any time from the results. The information is quite complete and gives a good depiction of the thermal performance of the die.

However, one disadvantage of transient solutions is that to obtain the thermal profile at quasi steady state literally hundreds of cycles are needed since the start-up temperature distribution is usually not close to the steady state and the die temperature will continue to change from cycle to cycle until the quasi-equilibrium is achieved. The multiple cycle simulation is time consuming and although it provides very complete information, some of the information is beyond that needed for the cycle and die cooling design since cycle and cooling design only requires the overall temperature distribution and part ejection temperature at steady state. Extra information is not bad, however the extra time required may limit exploration of "what-ifs." Thus it may be useful to have a tool for thermal analysis that provides the needed information and runs very quickly supporting interactive redesign and exploration of as many "what-ifs" as possible. The tradeoff is completeness of information with the time required for the analysis.

In quasi steady state, the temperature of a given point in the die may vary throughout the casting cycle depending on the distance from the cavity surface but the point returns to the same value at the same relative time in each cycle. In this work the equilibrium temperature at such a point is defined as the time average temperature over a cycle after the process reaches quasi steady state which is illustrated using Fig. 1and Fig. 2. Considering an arbitrary hypothetical point in the die, its temperature change over time is shown in Fig. 1. After a number of start-up cycles, cycle to cycle change is small and the process reaches the quasi steady state. The temperature change of this point over a single cycle thus can be illustrated using Fig. 2. The equilibrium and the transient temperatures for a die in quasi-equilibrium will be very close except for points relatively close to the cavity surface. Points on the cavity surface vary considerably in temperature over the cycle, but points in the interior of the die body do not.

Note that this is time average and not a spatial average so the temperature still varies spatially. This equilibrium temperature is helpful for cycle and cooling design, especially at conceptual design stage. A part of the research is to develop and implement a quick approach to compute the equilibrium temperature of die and part.

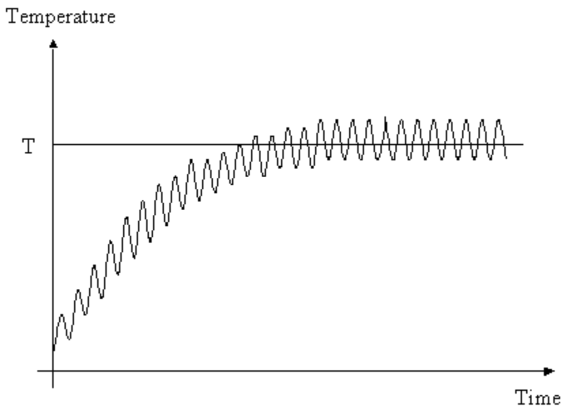


Fig. 1 Hypothetical temperature change of a point in the die over time.

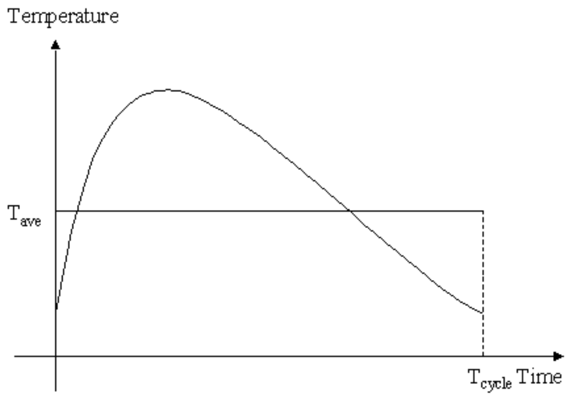


Fig. 2 Temperature change of the same point over a cycle in equilibrium state. At the end of a cycle, temperature comes back to the level at the beginning of the cycle.

The challenge in constructing equilibrium temperature calculations is two fold: 1) how to account for the heat released from the part without calculating the transient solution and 2) how to account for the changing conditions on the die interface. Both of these issues will be briefly reviewed. In addition, ideas about how to take advantage of rapid equilibrium calculations to support optimization of the die casting cycle and/or die cooling will be introduced.

Problem Formulation - Die

The voxel model used for other analyses in CastView is the basis for the equilibrium temperature calculation. Voxels represent the part and die as a set of small cubes of uniform size. For a given target voxel, there is a neighboring voxel at each face. The neighbors are denoted U, D, R, L, F, B for up, down, right, left, front, and back respectively.

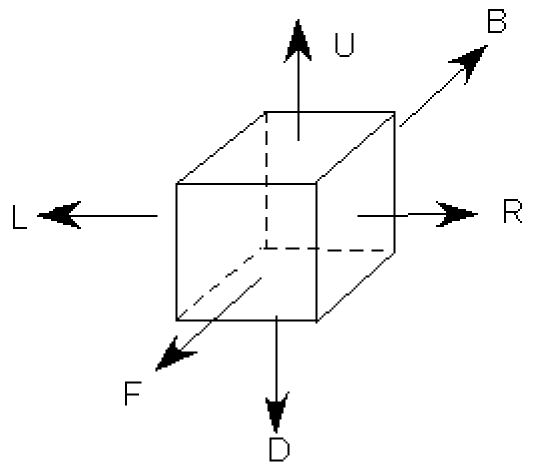


Figure 3 Voxel Relationships

For a die voxel, the heat transfer problem is addressed by analyzing the heat balance within a voxel considering the heat transfer that occurs with its neighbors across the 6 faces of the voxel. The rate of change of the temperature *T(t)* of a given voxel is proportional to the temperature difference with the surrounding voxels as shown in equation 1.

$$\frac{dT(t)}{dt}=\begin{Bmatrix} -\lambda_U\left(T(t)-T_U(t)\right)-\lambda_D\left(T(t)-T_D(t)\right)-\lambda_R\left(T(t)-T_R(t)\right) \\ -\lambda_L\left(T(t)-T_L(t)\right)-\lambda_F\left(T(t)-T_F(t)\right)-\lambda_B\left(T(t)-T_B(t)\right) \end{Bmatrix} \tag{1}$$

This is algebraically equivalent to equation 1a below which explicitly shows the dependence on the voxel temperature T and the weighted sum of the surrounding temperatures.

$$\frac{dT(t)}{dt}=\begin{Bmatrix} -\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)T(t) \\ +\lambda_U T_U(t)+\lambda_D T_D(t)+\lambda_R T_R(t)+\lambda_L T_L(t)+\lambda_F T_F(t)+\lambda_B T_B(t) \end{Bmatrix} \tag{1a}$$

Note that we cannot actually solve equation 1 or 1a by itself because of the spatial dependency (neighbors) that is present. The temperature of a given voxel depends on the temperature of its neighbors and the similarly for the neighbors. The neighbor interconnections create a large array of equations corresponding to the voxel array and the entire array must be solved together.

The proportionality constants are related to the thermal resistance and are dependent on the voxel size, material properties, and other parameters such has heat transfer coefficients. The various forms that the coefficients can take on are shown in table 1. The parameters referenced are ρ the density of the material in the target voxel, δ the dimension of one edge of the voxel, c_p the specific heat of the material in the voxel, k the conductivity of the voxel and the neighbor in those cases where there is dissimilar neighbor and h the interface heat transfer coefficient when there is an interface.

Table 1 Form of the Coefficients Determining Temperature Relationships

Case	**Coefficient** λ
Dissimilar Material (part to die, die to part)	$\frac{1}{\rho\delta c_p}\left(\frac{2k_1k_2h}{(k_1+k_2)h\delta+2k_1k_2}\right)$
Similar Materials (die to die – parting surface)	$\frac{1}{\rho\delta c_p}\left(\frac{kh}{h\delta+k}\right)$
Boundary, constant temperature boundary condition (part to air, die to air)	$\frac{1}{\rho\delta c_p}\left(\frac{2kh}{h\delta+2k}\right)$
Interior voxels, no interface	$\frac{k}{\rho\delta^2 c_p}$

Equilibrium conditions

Since we are interested in the equilibrium temperature, we integrate both sides of equation 1a to express the results in terms of averages. Integrating both sides between any two time points *t₀* and *t₁* gives

$$T(t_1)-T(t_0)=\left\{\begin{matrix}-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\overline{T} \\ +\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\end{matrix}\right\}\left(t_1-t_0\right) \qquad (2)$$

In words this equation says that the temperature difference between the start and end of the process is proportional to the difference between the average temperature of the voxel and its neighbors and the length of the time interval. Equivalently, the average rate of change is proportional to the difference in the average temperatures. If t_0 is the start of a cycle and t_1 is the end of the cycle, the temperature at these two time points is the same. In addition, if the neighboring voxels remain the same throughout the cycle (λ's constant), equation 2 reduces to

$$0=-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\overline{T}+\left(\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\right) \qquad (3)$$

or quite simply, the average temperature of a die voxel is the convex combination of the average temperature of its 6 neighbors.

$$\overline{T}=\frac{\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_L\overline{T}_L+\lambda_R\overline{T}_R+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B}{\lambda_U+\lambda_D+\lambda_L+\lambda_R+\lambda_F+\lambda_B} \qquad (4)$$

Equation 3 (or 4) applies to all die voxels except those on the parting surface and the cavity surface. The equation is exact for the situation that it addresses and is not an approximation, at least not within the assumptions used for its derivation. If a face adjoins a cooling line or is on the exterior exposed to air, the appropriate temperature boundary condition is applied and the unknown average temperature is replaced by the assumed temperature of the air or cooling line.

Techniques to solve systems of these equations were described in Wang and Miller (2002).

Surface/Interface Approximation

The voxels on the cavity surface and on the parting surface do not have the same neighbors throughout the cycle and therefore require an approximation. For purposes of calculation the cycle starts at the end of fill (t_{eof}) and proceeds through the subsequent stages as described in table 2. The heat transfer condition that applies at the interface will change from stage to stage and thus some of the coefficients in equation 1 will no longer be constants. For example, an ejector die cavity surface voxel in with one face in contact with the part will see heat transfer with the part during stages 0 and 1 and will then be exposed to air for stages 2, 4 and 5. Both the neighboring temperature and the coefficient for this face will be stage dependent.

Table 2 Definition of Cycle Stages.

Time Interval	**Cycle Stage Index**	**Definition**
$t_{eof} \le t < t_{open}$	0	Die closed and part in contact with both ejector and cover side. Part solidifying and cooling.
$t_{open} \le t < t_{ejected}$	1	Die open, prior to part ejection. Casting on the cover side is exposed to air. Casting in die contact on ejector side
$t_{ejected} \le t < t_{start}$	2	Parting surfaces and cavity surfaces exposed to air prior to spray
$t_{start} \le t < t_{stop}$	3	Spray occurs on designated surfaces.
$t_{stop} \le t < t_{cycle}$	4	All surfaces exposed to air prior to clamp
$t_{closed} \le t < t_{eof}$	5	Die closed. Parting surfaces are in contact. Cavity surfaces exposed to air inside the closed die cavity.

$$T(t_{eof}) - T(t_{open}) = \begin{Bmatrix} -\lambda_{U_0}\left(\overline{T}_0 - \overline{T}_{U_0}\right) - \lambda_{D_0}\left(\overline{T}_0 - \overline{T}_{D_0}\right) - \lambda_{R_0}\left(\overline{T}_0 - \overline{T}_{R_0}\right) \\ -\lambda_{L_0}\left(\overline{T}_0 - \overline{T}_{L_0}\right) - \lambda_{F_0}\left(\overline{T}_0 - \overline{T}_{F_0}\right) - \lambda_{B_0}\left(\overline{T}_0 - \overline{T}_{B_0}\right) \end{Bmatrix} \left(t_{eof} - t_{open}\right)$$

$$T(t_{open}) - T(t_{ejected}) = \begin{Bmatrix} -\lambda_{U_1}\left(\overline{T}_1 - \overline{T}_{U_1}\right) - \lambda_{D_1}\left(\overline{T}_1 - \overline{T}_{D_1}\right) - \lambda_{R_1}\left(\overline{T}_1 - \overline{T}_{R_1}\right) \\ -\lambda_{L_1}\left(\overline{T}_1 - \overline{T}_{L_1}\right) - \lambda_{F_1}\left(\overline{T}_1 - \overline{T}_{F_1}\right) - \lambda_{B_1}\left(\overline{T}_1 - \overline{T}_{B_1}\right) \end{Bmatrix} \left(t_{open} - t_{ejected}\right)$$

$$T(t_{ejected}) - T(t_{start}) = \begin{Bmatrix} -\lambda_{U_2}\left(\overline{T}_2 - \overline{T}_{U_2}\right) - \lambda_{D_2}\left(\overline{T}_2 - \overline{T}_{D_2}\right) - \lambda_{R_2}\left(\overline{T}_2 - \overline{T}_{R_2}\right) \\ -\lambda_{L_2}\left(\overline{T}_2 - \overline{T}_{L_2}\right) - \lambda_{F_2}\left(\overline{T}_2 - \overline{T}_{F_2}\right) - \lambda_{B_2}\left(\overline{T}_2 - \overline{T}_{B_2}\right) \end{Bmatrix} \left(t_{ejected} - t_{start}\right)$$

$$T(t_{start}) - T(t_{stop}) = \begin{Bmatrix} -\lambda_{U_3}\left(\overline{T}_3 - \overline{T}_{U_3}\right) - \lambda_{D_3}\left(\overline{T}_3 - \overline{T}_{D_3}\right) - \lambda_{R_3}\left(\overline{T}_3 - \overline{T}_{R_3}\right) \\ -\lambda_{L_3}\left(\overline{T}_3 - \overline{T}_{L_3}\right) - \lambda_{F_3}\left(\overline{T}_3 - \overline{T}_{F_3}\right) - \lambda_{B_3}\left(\overline{T}_3 - \overline{T}_{B_3}\right) \end{Bmatrix} \left(t_{start} - t_{stop}\right)$$

$$T(t_{stop}) - T(t_{closed}) = \begin{Bmatrix} -\lambda_{U_4}\left(\overline{T}_4 - \overline{T}_{U_4}\right) - \lambda_{D_4}\left(\overline{T}_4 - \overline{T}_{D_4}\right) - \lambda_{R_4}\left(\overline{T}_4 - \overline{T}_{R_4}\right) \\ -\lambda_{L_4}\left(\overline{T}_4 - \overline{T}_{L_4}\right) - \lambda_{F_4}\left(\overline{T}_4 - \overline{T}_{F_4}\right) - \lambda_{B_4}\left(\overline{T}_4 - \overline{T}_{B_4}\right) \end{Bmatrix} \left(t_{stop} - t_{closed}\right)$$

$$T(t_{closed}) - T(t_{eof}) = \begin{Bmatrix} -\lambda_{U_5}\left(\overline{T}_5 - \overline{T}_{U_5}\right) - \lambda_{D_5}\left(\overline{T}_5 - \overline{T}_{D_5}\right) - \lambda_{R_5}\left(\overline{T}_5 - \overline{T}_{R_5}\right) \\ -\lambda_{L_5}\left(\overline{T}_5 - \overline{T}_{L_5}\right) - \lambda_{F_5}\left(\overline{T}_5 - \overline{T}_{F_5}\right) - \lambda_{B_5}\left(\overline{T}_5 - \overline{T}_{B_5}\right) \end{Bmatrix} \left(t_{closed} - t_{eof}\right) \tag{5}$$

We are still considering the quasi-steady state so the temperature at the end of the cycle must equal the temperature at the start. The structure of the equations is such that the ending temperature at one stage is the starting temperature at the next and enters the equation with the opposite sign. Therefore, adding the equations together and collecting terms leads to

$$
\begin{aligned}
0 =& \\
&+\begin{bmatrix} -\left(\lambda_{U_0}+\lambda_{D_0}+\lambda_{R_0}+\lambda_{L_0}+\lambda_{F_0}+\lambda_{B_0}\right)\overline{T_0} \\ +\lambda_{U_0}\overline{T}_{U_0}+\lambda_{D_0}\overline{T}_{D_0}+\lambda_{R_0}\overline{T}_{R_0}+\lambda_{L_0}\overline{T}_{L_0}+\lambda_{F_0}\overline{T}_{F_0}+\lambda_{B_0}\overline{T}_{B_0} \end{bmatrix}\left(t_{eof}-t_{open}\right) \\
&+\begin{bmatrix} -\left(\lambda_{U_1}+\lambda_{D_1}+\lambda_{R_1}+\lambda_{L_1}+\lambda_{F_1}+\lambda_{B_1}\right)\overline{T_1} \\ +\lambda_{U_1}\overline{T}_{U_1}+\lambda_{D_1}\overline{T}_{D_1}+\lambda_{R_1}\overline{T}_{R_1}+\lambda_{L_1}\overline{T}_{L_1}+\lambda_{F_1}\overline{T}_{F_1}+\lambda_{B_1}\overline{T}_{B_1} \end{bmatrix}\left(t_{open}-t_{ejected}\right) \\
&+\begin{bmatrix} -\left(\lambda_{U_2}+\lambda_{D_2}+\lambda_{R_2}+\lambda_{L_2}+\lambda_{F_2}+\lambda_{B_2}\right)\overline{T_2} \\ +\lambda_{U_2}\overline{T}_{U_2}+\lambda_{D_2}\overline{T}_{D_2}+\lambda_{R_2}\overline{T}_{R_2}+\lambda_{L_2}\overline{T}_{L_2}+\lambda_{F_2}\overline{T}_{F_2}+\lambda_{B_2}\overline{T}_{B_2} \end{bmatrix}\left(t_{ejected}-t_{start}\right) \\
&+\begin{bmatrix} -\left(\lambda_{U_3}+\lambda_{D_3}+\lambda_{R_3}+\lambda_{L_3}+\lambda_{F_3}+\lambda_{B_3}\right)\overline{T_3} \\ +\lambda_{U_3}\overline{T}_{U_3}+\lambda_{D_3}\overline{T}_{D_3}+\lambda_{R_3}\overline{T}_{R_3}+\lambda_{L_3}\overline{T}_{L_3}+\lambda_{F_3}\overline{T}_{F_3}+\lambda_{B_3}\overline{T}_{B_3} \end{bmatrix}\left(t_{start}-t_{stop}\right) \\
&+\begin{bmatrix} -\left(\lambda_{U_4}+\lambda_{D_4}+\lambda_{R_4}+\lambda_{L_4}+\lambda_{F_4}+\lambda_{B_4}\right)\overline{T_4} \\ +\lambda_{U_4}\overline{T}_{U_4}+\lambda_{D_4}\overline{T}_{D_4}+\lambda_{R_4}\overline{T}_{R_4}+\lambda_{L_4}\overline{T}_{L_4}+\lambda_{F_4}\overline{T}_{F_4}+\lambda_{B_4}\overline{T}_{B_4} \end{bmatrix}\left(t_{stop}-t_{closed}\right) \\
&+\begin{bmatrix} -\left(\lambda_{U_5}+\lambda_{D_5}+\lambda_{R_5}+\lambda_{L_5}+\lambda_{F_5}+\lambda_{B_5}\right)\overline{T_5} \\ +\lambda_{U_5}\overline{T}_{U_5}+\lambda_{D_5}\overline{T}_{D_5}+\lambda_{R_5}\overline{T}_{R_5}+\lambda_{L_5}\overline{T}_{L_5}+\lambda_{F_5}\overline{T}_{F_5}+\lambda_{B_5}\overline{T}_{B_5} \end{bmatrix}\left(t_{closed}-t_{eof}\right)
\end{aligned}
\qquad (6)
$$

Problem Formulation - Part

The equations for a part voxel are similar but a little more complex due to the fact that the part has latent heat. Following a similar path it can be shown that the average temperature relationship for the part is as follows:

$$
T_{eject}-T_{inj}=\frac{L}{c_p}f_s+\left\{\begin{matrix} -\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\overline{T} \\ +\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B \end{matrix}\right\}\left(t_{eject}-t_{inj}\right) \qquad (7)
$$

where L is the latent heat, c_p is the specific heat and f_s is the fraction solid at the time of ejection. The equation shows that the difference between the injection temperature and the ejection temperature of the part is proportional to the weighted average temperature difference plus a term proportional to the fraction of the latent heat released. A simple linear approximation is used for the latent heat released so the f_s term is a function of the ejection temperature in this formulation. Equation 7 was derived under the assumption that the interface between the part and the die is constant until ejection. This is true on the ejector side of the die, but not the cover. Cavity surface voxels on the cover side are exposed to the die until the die opens and then exposed to air until the part is ejected. This requires splitting the time interval into two stages as was done in the previous section for the die and developing an approximation. We ignore the details for this discussion.

Clearly equation 7 describes the average part temperature in terms of the average temperature of neighboring voxels and heat sources including the latent heat and sensible heat. For part surface voxels, at least some of the neighboring voxels will be die voxels and the heat released by the part becomes the heat input to the die. Unlike the die, the part is not in equilibrium so the temperature difference does not disappear. We must determine of estimate the ejection temperature as a function of available information.

Ejection Temperature

Motivation for the approach taken to approximate the ejection temperature is provided by the data in figure 5 which was generated via numerical simulation of the cooling of the part shown in figure 4 using Abaqus. Note that these data are produced with a transient simulation, not an equilibrium calculation.

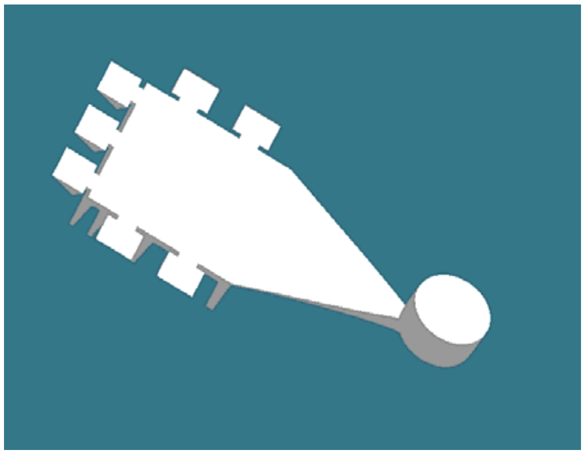


Figure 4 – Test Part

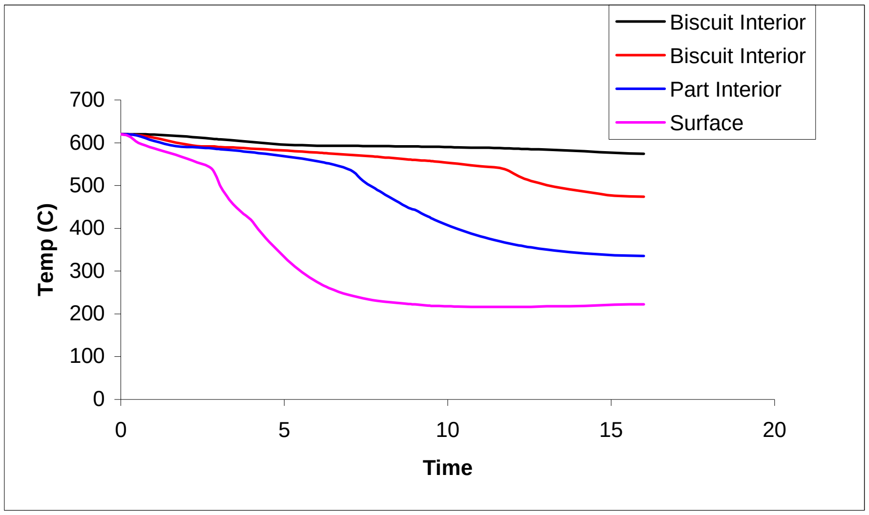


Figure 5. Cooling curves for 4 points from the flat plate part

The casting alloy used for the analysis is A380 and the injection temperature is assumed to be 620^{o} C. Liquidus is about 593 ^{o}C. The black line in figure 5 was taken from a point on the interior of the biscuit, a point among the slowest to cool and solidify. The red line was also from the biscuit interior but much closer to the surface. The blue line is from the interior of the plate and the magenta line is near the tip of one of the overflows.

The explanation for the type of behavior exhibited in figure 5 is provided in part by figure 6. The figure displays curves for 4 points on the part surface and 4 mating points on the die surface. The part curves show that the part is loosing temperature relatively quickly, particularly once the latent heat is removed. Conversely, the die is heating up until at about the 9 or 10 second mark, the two temperatures are very close together. From this point onward, the part cooling rate decreases significantly because of the small temperature differential with the die. It is within this region that the part temperature curve appears to asymptote. This is really not an asymptote but an indication that the cooling is now controlled by dissipation of heat by the die, a slower process than the initial dumping of heat from the part.

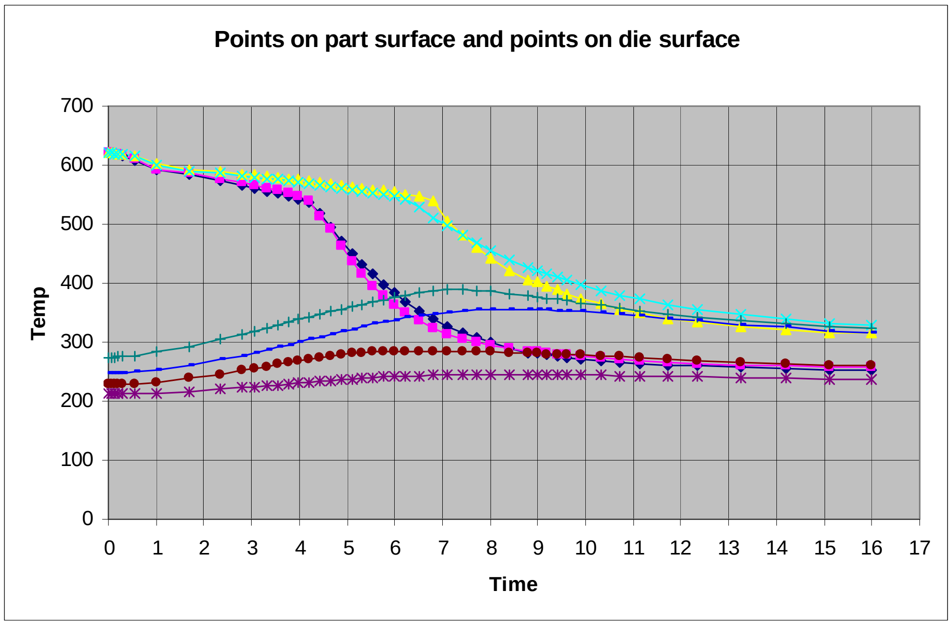


Figure 6. Comparison of adjacent die and part temperature curves

The shape of the curves has a distinct linear response characteristic, but the response is different for each location. In those areas where the die has a higher temperature, the part temperature changes a little more slowly. Where the die temperature is lower the part cools more. We take advantage of this fact to try to approximate the relationships between local conditions and the part ejection temperature.

If we didn't know what the source of the data in figures 5 and 6 was, we might guess that the data are generated by a simple linear system and we will use such a system as a surrogate representation of the more complicated actual case. The surrogate that we will use is

$$\frac{dz(t)}{dt} = \alpha(z(t))\left(-\gamma z(t) + \gamma T_c\right);\; z(t_0) = T_{inj} \qquad (8)$$

where z(t) is the surrogate temperature, γ is an unknown rate constant and T_c is an unknown constant forcing function. The remaining term $\alpha(z(t))$ decreases the rate of change in the mushy zone and is defined as follows

$$\alpha(z(t)) = 1$$

outside of the mushy zone and

$$\frac{c_p\left(T_{liq} - T_{sol}\right)}{c_p\left(T_{liq} - T_{sol}\right) + L} < 1$$ inside. The effect is to decrease the rate of change while the latent heat is released.

The surrogate will be used to calculate an ejection temperature analytically as a function of the various parameters and to calculate an average temperature also. The surrogate values will then be used instead of the values determined by the equilibrium equations.

Equation 8 outside of the mushy zone is very simple,

$$\frac{dz(t)}{dt} = -\gamma z(t) + \gamma T_c = -\gamma \left(z(t) - T_c \right) \qquad (9)$$

The rate of change is proportional to the difference between z and T_c. Inside the zone, the form is identical but the magnitude of γ is reduced as described above so that the temperature changes much more slowly. Basically this means that we have 3 stages of response. First, the rate of removal of superheat is described by γ the difference between the injection temperature and Tc. The response in this stage is nearly linear with time. Second, during removal of the latent heat the temperature changes much more slowly due to the attenuation factor mentioned above. Finally, finally part cooling is again described by equation 9 and the rate of cooling will gradually decrease as z moves closer to T_c. In a nutshell, we have an equation that behaves approximately like the actual situation and a constant forcing term T_c captures the dependence on the local neighbors.

The average relationship for the surrogate is very similar to equation 7 and is obtained by integrating equation (9) between injection and ejection. The result is

$$z_{eject} - T_{inj} = \frac{L}{c_p} f_s - \gamma \left(t_{eject} - t_{inj} \right) \left(\bar{z} - T_c \right) \qquad (10)$$

The advantage of the surrogate equation is that we can solve it quite readily. The general procedure is based on the fact that the solution to the surrogate is of the form (α = 1 for the moment)

$$z(t) - T_c = e^{-\gamma(t - t_0)} \left(T_{inj} - T_c \right) \qquad (11)$$

The ejection temperature can be above liquidus, in the mushy zone, or below the solidus. Each case has to be taken separately, but we will consider only the case where ejection is below solidus for the development. Using equation 10 with the boundary conditions applicable for each phase, the solution pieces can be expressed in terms of the unknown points in time at which the system enters and leaves the mushy zone, ie.

$$T_{liq} - T_c = e^{-\gamma(t_{sh} - t_0)} \left(T_{inj} - T_c \right)$$
$$T_{sol} - T_c = e^{-\alpha\gamma(t_{lh} - t_{sh})} \left(T_{liq} - T_c \right)$$
$$z_{eject} - T_c = e^{-\gamma(t_{eject} - t_{lh})} \left(T_{sol} - T_c \right)$$

Taking the natural log of each equation, adding the results together, and doing just a little algebra leads to

$$-\gamma(t_{eject} - t_0) = \ln\left[\frac{z_{eject} - T_c}{T_{inj} - T_c} \right] + \frac{L}{c_p \left(T_{liq} - T_{sol} \right)} \ln\left[\frac{T_{sol} - T_c}{T_{liq} - T_c} \right] \qquad (12)$$

Equation 10 then becomes

$$z_{eject} - T_{inj} - \frac{L}{c_p} f_s = \left\{ \ln\left[\frac{z_{eject} - T_c}{T_{inj} - T_c} \right] + \frac{L}{c_p \left(T_{liq} - T_{sol} \right)} \ln\left[\frac{T_{sol} - T_c}{T_{liq} - T_c} \right] \right\} \left(\bar{z} - T_c \right) \qquad (12)$$

$$\bar{z} = T_c + \frac{z_{eject} - T_{inj} - \frac{L}{c_p} f_s}{\ln\left[\frac{z_{eject} - T_c}{T_{inj} - T_c} \right] + \frac{L}{c_p \left(T_{liq} - T_{sol} \right)} \ln\left[\frac{T_{sol} - T_c}{T_{liq} - T_c} \right]}$$

Note the similarity with equation 7 and note how T_c replaces the neighbor dependencies. Now, if the surrogate and actual match with respect to average and ejection temperature, ie $\overline{z}=\overline{T}$ and $z_{eject}=T_{eject}$, from 7 and 10 we find that T_c must satisfy

$$\overline{T}-T_c=\frac{\left(t_{eject}-t_{inj}\right)}{-\gamma\left(t_{eject}-t_{inj}\right)}\left\{\begin{array}{l}-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\overline{T}\\+\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\end{array}\right\}\qquad(11)$$

and similarly equation 11 becomes

$$0=T_c-\overline{T}+\frac{\left(t_{eject}-t_{inj}\right)\left\{\begin{array}{l}-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\overline{T}\\+\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\end{array}\right\}}{\ln\left[\dfrac{z_{eject}-T_c}{T_{inj}-T_c}\right]+\dfrac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\dfrac{T_{sol}-T_c}{T_{liq}-T_c}\right]}\qquad(13)$$

$$\overline{T}=\frac{\left[\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\right]}{\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)}-\frac{T_{eject}-T_{inj}-\dfrac{L}{c_p}f_s}{\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)}$$

$$\overline{z}=T_c+\frac{z_{eject}-T_{inj}-\dfrac{L}{c_p}f_s}{\ln\left[\dfrac{z_{eject}-T_c}{T_{inj}-T_c}\right]+\dfrac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\dfrac{T_{sol}-T_c}{T_{liq}-T_c}\right]}$$

$$T_c=\frac{\left[\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\right]}{\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)}$$

$$\frac{T_{eject}-T_{inj}-\dfrac{L}{c_p}f_s}{-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)}=\frac{z_{eject}-T_{inj}-\dfrac{L}{c_p}f_s}{\ln\left[\dfrac{z_{eject}-T_c}{T_{inj}-T_c}\right]+\dfrac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\dfrac{T_{sol}-T_c}{T_{liq}-T_c}\right]}$$

$$\frac{T_{eject}-T_{inj}-\frac{L}{c_p}f_s}{z_{eject}-T_{inj}-\frac{L}{c_p}f_s}=\frac{-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)}{\ln\left[\frac{z_{eject}-T_c}{T_{inj}-T_c}\right]+\frac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\frac{T_{sol}-T_c}{T_{liq}-T_c}\right]}$$

$$\ln\left[\frac{z_{eject}-T_c}{T_{inj}-T_c}\right]+\frac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\frac{T_{sol}-T_c}{T_{liq}-T_c}\right]=-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)$$

$$T_c=\frac{\left[\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\right]}{\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)}$$

$$\ln\left[\frac{z_{eject}-T_c}{T_{inj}-T_c}\right]=-\frac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\frac{T_{sol}-T_c}{T_{liq}-T_c}\right]-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)$$

$$\ln\left[\frac{z_{eject}-T_c}{T_{inj}-T_c}\right]=\ln\left\{\left[\left[\frac{T_{sol}-T_c}{T_{liq}-T_c}\right]^{-\frac{L}{c_p\left(T_{liq}-T_{sol}\right)}}\right]e^{-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)}\right\}$$

$$\frac{z_{eject}-T_c}{T_{inj}-T_c}=\left[\left[\frac{T_{sol}-T_c}{T_{liq}-T_c}\right]^{-\frac{L}{c_p\left(T_{liq}-T_{sol}\right)}}\right]e^{-\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)}$$

Our approximate solution is achieved if equations 7, 12 and 13 are satisfied simultaneously. Equations 12 and 13 are highly nonlinear and cannot be solved in closed form but can be solved using search techniques.

$$T_c+\frac{z_{eject}-T_{inj}-\frac{L}{c_p}f_s}{\ln\left[\frac{z_{eject}-T_c}{T_{inj}-T_c}\right]+\frac{L}{c_p\left(T_{liq}-T_{sol}\right)}\ln\left[\frac{T_{sol}-T_c}{T_{liq}-T_c}\right]}$$

$$=\frac{\left[\lambda_U\overline{T}_U+\lambda_D\overline{T}_D+\lambda_R\overline{T}_R+\lambda_L\overline{T}_L+\lambda_F\overline{T}_F+\lambda_B\overline{T}_B\right]}{\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)}-\frac{T_{eject}-T_{inj}-\frac{L}{c_p}f_s}{\left(\lambda_U+\lambda_D+\lambda_R+\lambda_L+\lambda_F+\lambda_B\right)\left(t_{eject}-t_{inj}\right)}$$

Illustration of the Surrogate

Figure 7 is a computational demonstration that the surrogate approach works quite well. This is not an equilibrium calculation, but a direct test of the degree to which the first order surrogate matches the cooling curves developed from a numerical simulation.

The procedure used is as follows:

- A table of average z values was computed as a function of z_{eject} and T_c using equation 10.
- Given the ejection temperature and the average temperature from each curve, a value for T_c was determined using the table.
- With values for T_c and z_{eject} available, γ was computed from equation 11.
- The time response was then computed from the solutions of the differential equation.

The match is excellent suggesting that the surrogate is a good approximation in the sense that the T_c does a good job approximating the neighbors and the combination of parameters captures the time response quite well. The surrogate relationships therefore can be used to relate the average temperature with the ejection temperature as required.

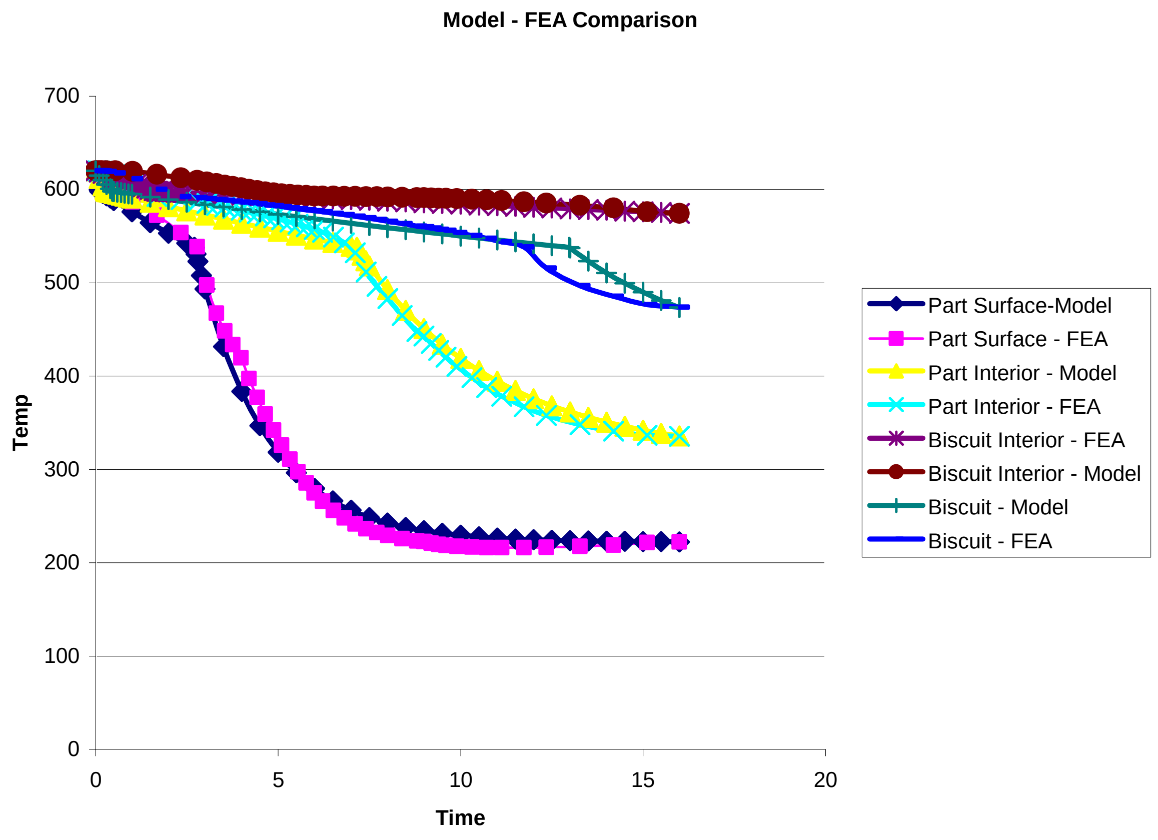


Figure 7 Comparison of Surrogate with Numerical Simulation

Example Equilibrium Solution

Several illustrations of the results of the equilibrium calculations are shown below. Figures 8 and 9 show both sides of the part and display ejection temperature. The hot spot is the biscuit as would be expected with the center of the plate the second hottest region. The corresponding average temperatures for the inserts are shown in Figures 10 and 11. The hottest region for the die is in the area between the ribs at an average temperature of a little under 400 degrees C.

The cooling line sketcher function was used to add 3 cooling lines, one near the biscuit and two in the larger rib spacings. The layout is shown in figure 12. The red lines correspond to the center lines of the cooling line. The cooling lines were set to be approximately ½” in diameter carrying 30 degree C water at a flow rate of 2 gallons per minute. The resulting part ejection temperature and average ejector insert temperature are shown in figures 13 and 14. The peak temperatures are reduced with cooling as expected and both the part and die temperature distributions are affected. It is the ability to manipulate these distributions that we are concerned with. The effect of the cooling lines is clearly evident in the waviness of the blue pattern near the surface of the insert as shown in figure 14.

The cooling line effects are even more visible in figures 15 and 16 which show the insert surface temperature with and without cooling lines respectively. The color scales are not the same in the two figures. The dark blue circular regions in figure 16 correspond to the locations of the cooling line. It is very clear that the cooling lines change the temperature pattern but it is also equally clear, particularly from figure 16, that the cooling lines’ impact rapidly falls off as you move away from the line. The effect is somewhat local. The lines do however lower the overall die temperature given the same cycle time. This means that the cycle time might be reduced a little if the biscuit can be solidified sufficiently.

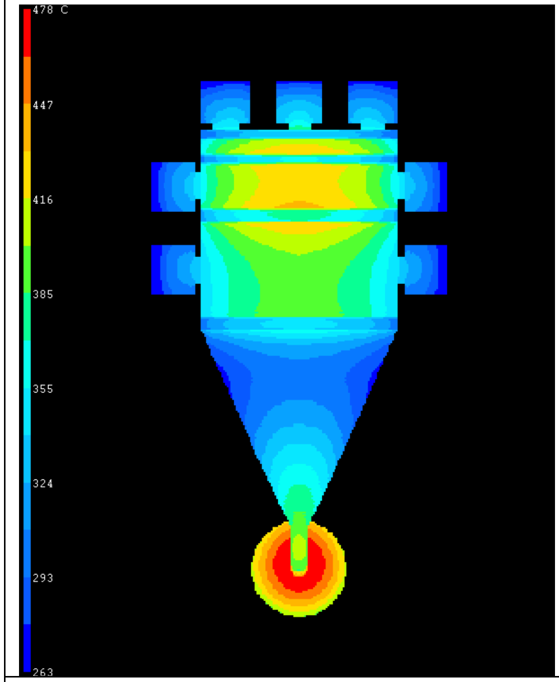


Figure 8 - Ejector side of the part

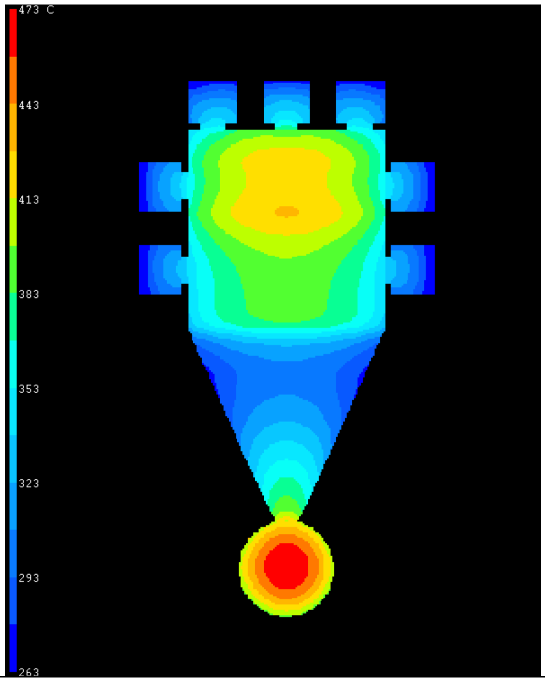


Figure 9 - Cover side of the part

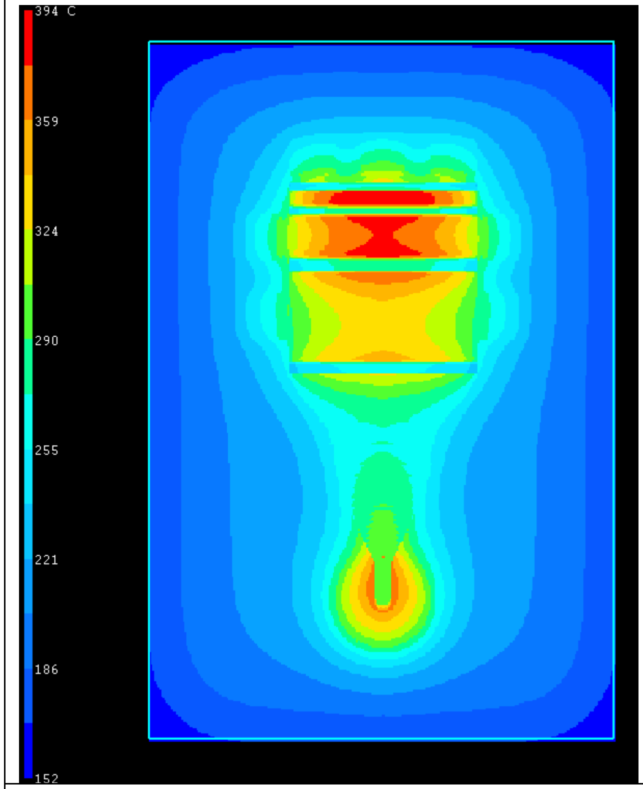
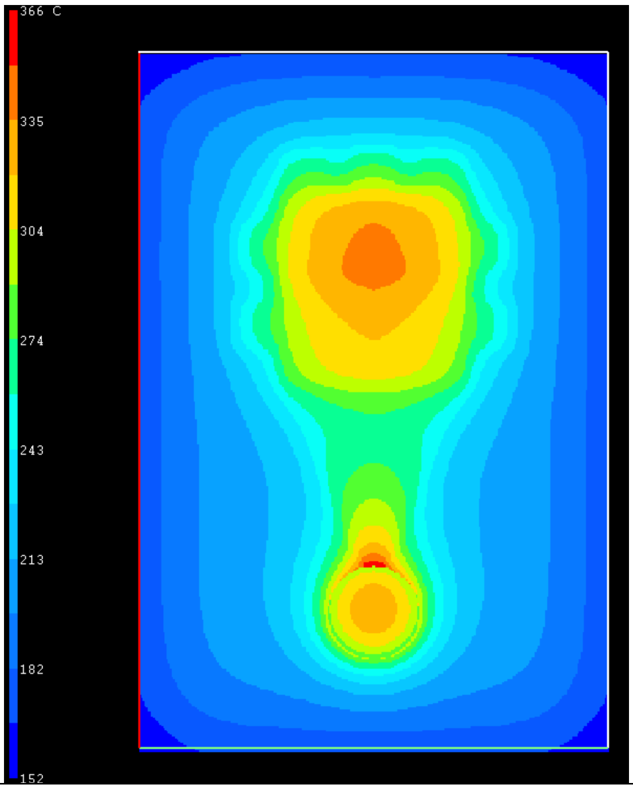
	

Figure 10 Ejector Insert – Average Temperature	Figure 11 Cover Insert – Average Temperature

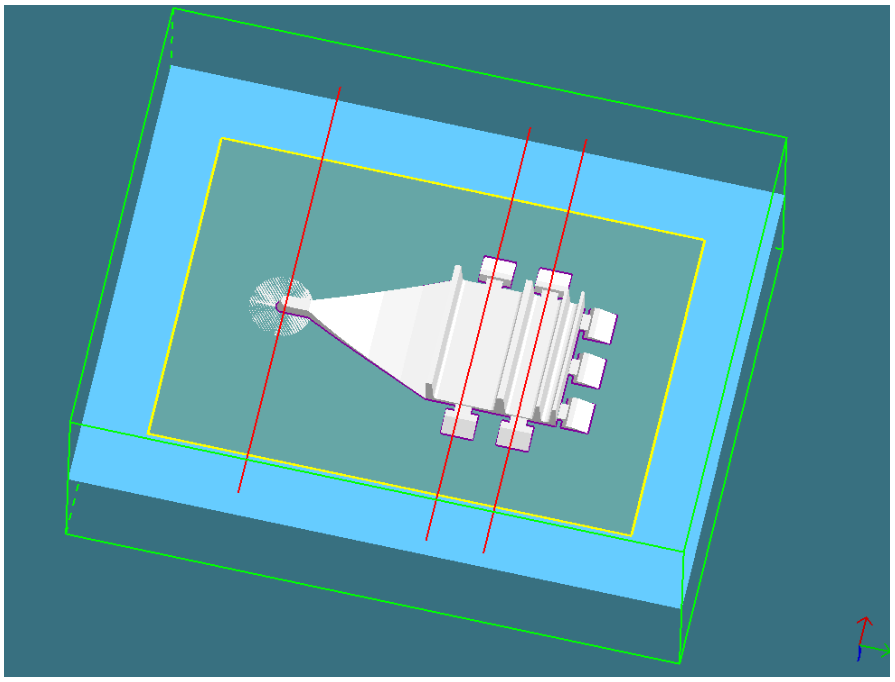


Figure 12 Cooling line layouts

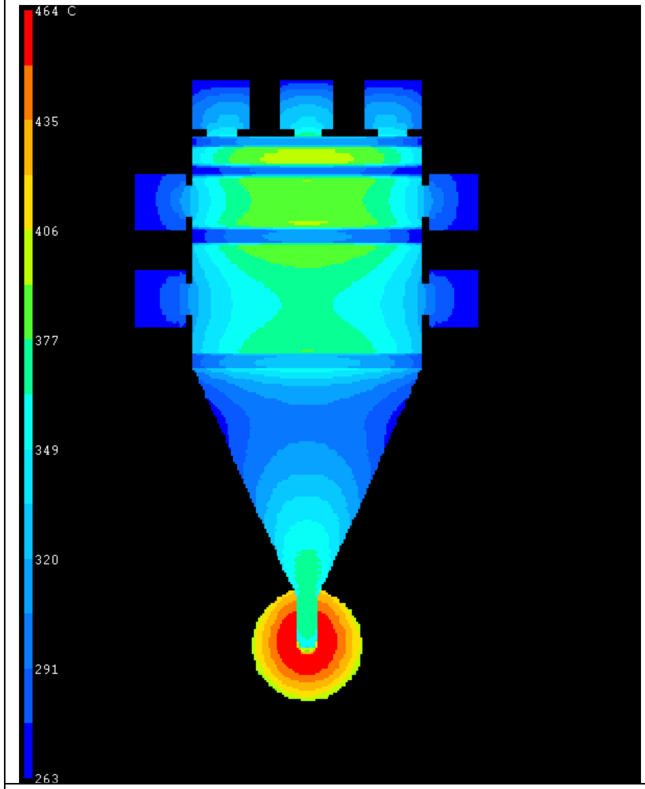


Figure 13 Part Ejection Temperature with Cooling

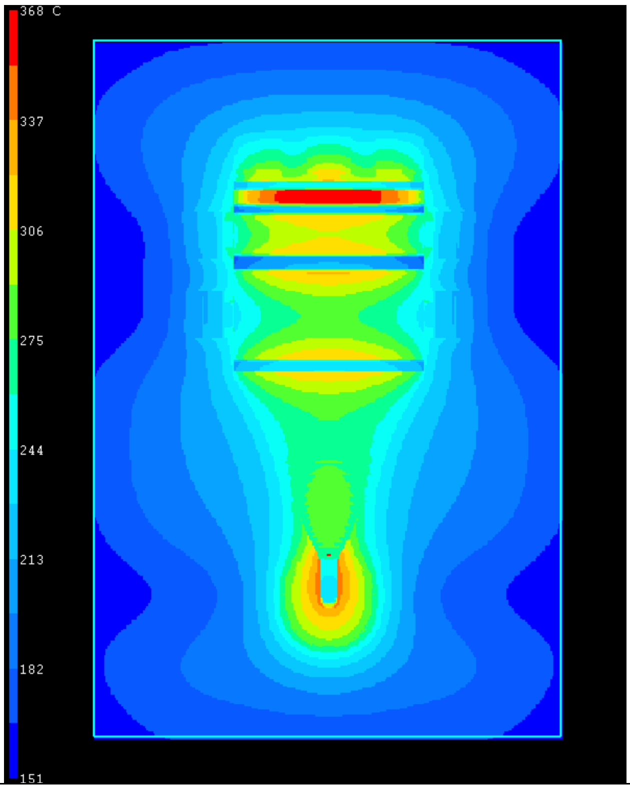


Figure 14 Ejector Insert Average Temperature

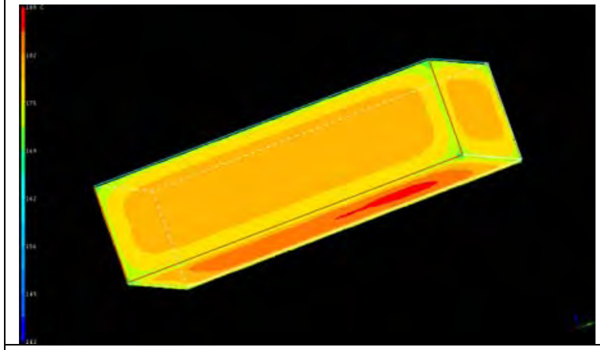


Figure 15 – Insert Surface Temperatures – No Cooling Lines

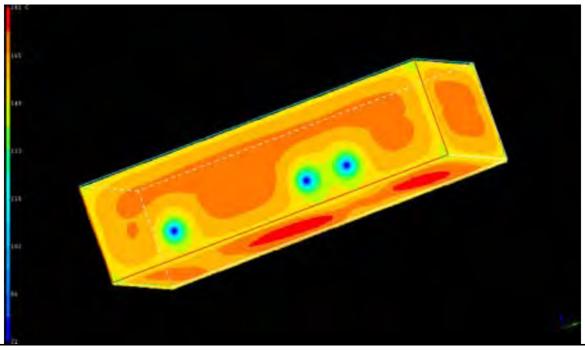


Figure 16 – Insert Surface Temperatures – With Cooling Lines

Future Work – Optimization

The time required to compute these temperature patterns at a 200 voxel resolution is a about 150 seconds on a 2.8 Ghz machine. No changes

References

Dongtao Wang, R. Allen Miller and Li Cai, “Equilibrium Temperature Analysis for Die and Part”, *Proceedings of 22nd International Die Casting Congress and Exhibition*, Chicago, 2002.

1. Allchin, T., Quality assurance with modern CNC die casting system, *Die Casting Engineer*, 1990, March/April, p34-38.

2. Barone, M. R. and Caulk, D. A., dieCas--Thermal analysis software for die casting: modeling approach, *Proceeding of 17th International Die Casting Congress and Exposition*, 1993.

3. Barone, M. R. and Caulk, D. A., Analysis of liquid metal flow in die casting, *International Journal of Engineering Science*, 2000, V38, p1279-1302.

Appendix F: Automated Geometric Modeling of Die Components

Automated Geometric Modeling of Die Components

Li Shen

Center for Die Casting
The Ohio State University

June 2007

1. Introduction

1.1 Die Casting Dies

Die casting dies are metal molds, composed of an ejector die and a cover die. The ejector die is attached to movable platen of the casting machine. The cover die is located on the non-movable side towards the molten metal injection system. Because of the high pressure and high temperature work environment, dies are typically made out of alloy steel. It must have higher melting point than the metal material it will contain, otherwise it will be damaged or distorted [3]. The cover die half contains cavity that is responsible for forming the top half of a part. In hot chamber machines, the cover die holds the sprue through which molten metal enters the die. The ejector half is designed to contain the ejector mechanism and, in most cases, the runners and gates that route molten metal to the cavity (or cavities) of the die [4]. The pair of opposite directions along which the two die halves separate are the die opening directions [4]. Die cores, either fixed or movable, may be placed in either die half. These allow holes to be cast in various directions and various inserts may be pre-positioned in the dies to become integral casting features [4]. Cooling system is embedded in dies to speed up the cooling stage of a formed part. The die cavity into which the casting part is formed is machined into both halves of the die block or into inserts that are secured in the die blocks [4]. After the metal solidifies and the die opens, ejector pins push the casting from the ejector die half. Before closing for the next injection cycle, the dies are subjected to an air blast to be cleaned, and then given a lubricating spray [4].

1.2 Problem Statement

Based on the fact that die casting process is a complex operation involving a variety of variables, such as material characteristics, pressure, fluid velocity, etc., each of which have an impact on the casting quality, die configuration and design requires expertise and often long cycle times from design to actual manufacturing. As computer science and computer aided design and manufacturing (CAD/CAM) rapidly developed during the last two decades, advanced software and tools have been implemented in die casting industry trying to help manufacturer better design and produce castings. However, most of the CAD/CAM tools utilize numerical simulation. Numerical simulation is founded on complex physical mathematical model. A collection of process data (including casting and gating system geometrical characteristics, material thermal properties and process parameters) is input into the computation functions, and then the simulation program calculates related data and output simulation results [5].

Numerical simulation offers a powerful way to analyze complex metal flows and to optimize the design of the gating system and the geometry of the dies, but many design decisions must be made prior to executing the simulation [5]. In addition, at the conceptual design stage when designers only care about making quick decisions among many alternative designs, a faster-response system is helpful to visualize alternative ideas and remove poor designs.

A structured geometric modeling approach is a desirable way to construct conceptual die design and reasoning. Though the analysis accuracy of geometric analysis is lower than that of numerical simulation, it is adequate to provide designers with qualitative analysis results during conceptual design stage. The beauty of geometric method is the quickness to obtain the analysis results. In addition, since the thermal profile, fill pattern and other process features are closely related to the geometry of casting and die, it is possible to obtain the feature information based on the pure geometric model of dies for the purpose of future flow pattern and thermal analysis.

1.3 Literature review

Die system modeling and simulation was adopted very extensively in the early 1980s with simple die characteristics calculation and solidification simulation packages. The rest of the section will discuss some of the most used die modeling methods.

1.3.1 Feature Based Parametric Die Design

Traditionally, parametric and feature based design are two separated modeling approaches. The former method deals with variable dimensions as control parameters, and it is an effective tool for creating models based on parameters [7]. A parametric model that contains the principle shape and parameters for the dimensions, is first defined and stored [8]. Any variation of the parametric model, which has the same topological constraints but different dimensions can be obtained by assigning a set of specific values to the parameters of the parametric model [9]. This approach allows a parametric model to represent a part family whose members are different in dimensions but identical in topology [10]. It is used to design a product with features that are functionally defined by attributes and are geometrically represented by a set of parameters.

Feature-based design is used to design a product with features that are functionally defined by attributes and are geometrically represented by a set of parameters [12]. Technically parametric design is the foundation of feature-based design. Besides inheriting all the advantages of parametric design, it contains a library of features for general design purposes [7]. Some CAD software provides such standard features as *hole*, *slot*, *cylinder*, *cone*, etc, which are adequate for wide-ranging needs.

In some applications, user-defined features are allowed to provide special modeling needs. Besides standard shapes, irregular shapes, which can be defined and displayed by specifying critical parameters, can be generated by users and stored in the data library for future use. Design tool with this type of feature is called feature-based and parametric design. There have been some studies on designing individual die elements by using feature-based methodology. For example, Woon et al. [6] developed a prototype design of die system. However, it only provided a simple frame for die geometry and focused more on the gating system for a die-casting using a P-Q^2 technique and feature-based parametric design rather than die geometric modeling. As for whole die design, Choi et al. [11] developed a similar system based on the AutoCAD platform. However, the system was only applicable for simple shapes such as the cap-shape. Moreover, the system could only be used for products that have no undercuts. Chen et al. [12] also did a study on an integrated CAD/CAE system for die casting. The idea was to determine the die geometry and process parameters based on feature parameters and then to use the CAE package to optimize the process design based on the simulation analysis. To take the research of die design system further, Y.K Woon et al. [6] imported more modeling functions into the feature parametric based method. In their research, the modeling procedure was divided into three steps: (1) feature-based and constraint-based modeling, (2) parametric design and (3) geometry and topological extraction techniques from boundary representation.

However, all the methods mentioned above can offer automated generation of either gating system or the whole die system, but they are all restricted by the shape of casting parts. None of them is sufficient for handling complex shaped castings.

1.3.2 Case-based Die Design

Since every designed part has its unique die geometry and gating system, it is extremely time consuming when casters have to work with multiple parts. Many researchers have being trying to design an expert system for die design, which contains artificial intelligence knowledge library in its system, eliminates repeated work, and supports reusability in die design.

However, there are certain limits to this method, such as no simple rules can cover the knowledge and experience in die casting die design and difficulties to organize and recognize irregular features. Based on the concept of expert system, recent years, a case-based reasoning (CBR) methodology [13] has been proposed. A case recorded not only the result, but also the design concept and intention, such as in what situation, according to what conditions and how this result was obtained. A CBR system adapted the solution of a previously solved case to build a solution for a new problem. This is a more useful method than the use of an expert system to simulate human thought because proposing a similar case and applying a few modifications seems to be self-explanatory and more intuitive to many humans [1]. Various case-based design support tools have been developed for numerous tasks such as in injection molding design [14], architectural design [15, 16], fixture design [17] and process planning [18]. K. S. Lee et al. [1] introduced a cased-based die design system derived from latest parametric design technology. Each case was compared with case database by searching through the feature parameters of die. If there were similarities, basic structure of die will be constructed based on old case and new features can be added onto the old die design. B. Humm et al. [18] proposed a similar case-based system. This system more focused on die casting process design. While the implementation of a case-based system is limited due to the inevitable disadvantage of creating accurate indexes of similar cases, complete index rules and capturing important feature parameters. In addition, since it is based on feature parameters, it will run into troubles when dealing with irregular shaped parts.

1.4 Research Objective

A die system can be grouped into five segments: (1) filling system including gates, sprues, runner, overflow and venting (2) functional components including cavity, insert and slides. (3) ejector system, (4) cooling system, (5) standard components, such as die base, blocks etc. As discussed in previous section, there are different approaches being implemented in die casting die design field. Most of the researches were dedicated to the design of the filling system and cooling system. On the other hand, for research or software in which the geometric modeling of functional components is considered, there are modeling limits at different level. Hence, the ultimate goal is to design a CAD tool that allows users easily create a precise and accurate 3D model of die with only the geometric data of part available. The tool should be able to automatically identify special features, such as undercuts and holes. The detailed research objectives are listed as below:

A. The tool should provide accurate and precise configuration of dies' functional components and represent the results as a 3D model.
B. The new generated models should have compatible data structure with existing data structure in the system.
C. Data structure should be simple and easy for future operations on the data, such as thermal analysis.
D. User interface should be easy for use and requires as little prerequisite information about die system as possible.

In following section, the methodology for constructing dies' functional components is discussed and any necessary tools or interface is provided. First, the modeling process starts with the dexelization calculation (details about dexelization will be introduced in chapter 3), taking the geometric model and predefined die dimension as inputs. Second, all triangles, which are the fundamental elements of the part geometric model, are classified into different groups based on their location and relationship to the cover die, ejector die and parting surface. Third, special features are identified and related triangles are reclassified. Fourth, new surfaces corresponding to special features are constructed and retriangulated.

2. Model Structure

2.1 Introduction

Stereolithography (STL) file format is very commonly used for data interchange because of its consistent, CAD independent data structure and capability of modeling complex shaped 3D models.

An STL model uses triangles to approximate the actual surface of a part as shown in Figure 2.1 . Three vertices and a unit normal uniquely identify each triangle of the STL model. Most current CAD systems have the capability of creating STL files. The STL model is also the basic data structure for representing the part model in CastView.

The surface of a die cavity has the same surface features as its part model. Therefore, by reasoning the surface of a part model, information related to die cavity can be retrieved. If user can identify which triangles on the part surface are adjacent to the cover die or the ejector die, they can directly use these correlations to build die models. However, under the condition of only having the part's STL model on hand, every triangle element has the same characteristics. Without other information, it would not be possible to tell which triangles belong to the cover die and which ones belong to the ejector die. To solve this problem, dexel model is imported as an intermediate data structure before die models are created. Dexel model is composed of line segments, which are parallel to the die open direction. In this research, it is first used to classify triangles, which constitute the surface of a part. Then it is utilized to identify special features, such as undercuts and holes.

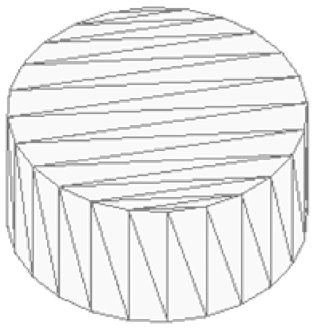


Figure 2.1 Example of STL model

After dexelization, triangles from the part model, which require no further work, are directly copied to the dies' STL files with reversed normal. For the rest of the triangles, they are either retriangulated or used to build new surfaces. All new generated triangles are inserted in one of the two dies STL triangle lists with topology information to construct complete and accurate die models.

2.2 STL Data Structure

STL model was adopted in this research to represent final die models. Simple STL data only store triangles in a linked list (Figure 2.2). Each triangle element contains three vertices and the normal of the triangle.

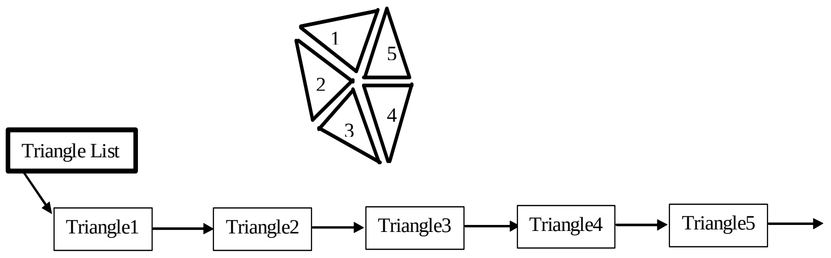


Figure 2.2 Simple STL data structure

However, the topological information is not stored and there are many redundant data. Therefore, a structured STL data structure was created. It can provide not only the coordinates of each vertex but also the topology information [21]. It has two basic segments. First is the vertex node and the second one is the triangle node. A vertex node contains coordinates of the vertex, the pointer to the next vertex node and a triangle list of all the triangles that share the same vertex (Figure 2.3). A triangle node is based upon vertex nodes and is the fundamental element for 3D display. A triangle node contains pointers to 3 vertex nodes, pointers to 3 neighbors and the pointer to the next triangle node (Figure 2.3). Each triangle node is stored in a triangle list and every vertex node is stored in a vertex list (Figure 3.4).

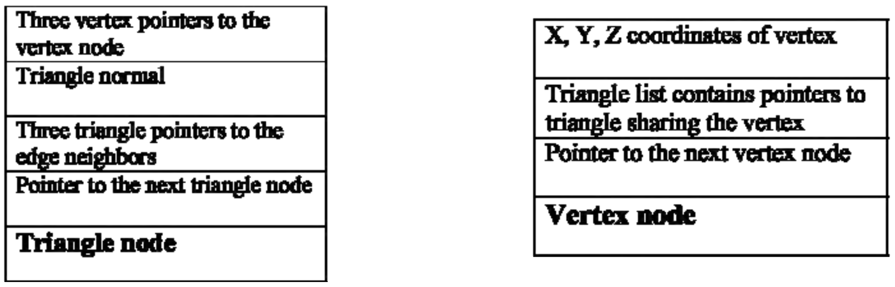

Three vertex pointers to the vertex node
Triangle normal
Three triangle pointers to the edge neighbors
Pointer to the next triangle node
Triangle node
X, Y, Z coordinates of vertex
Triangle list contains pointers to triangle sharing the vertex
Pointer to the next vertex node
Vertex node

Figure 2.3 STL data structure [21]

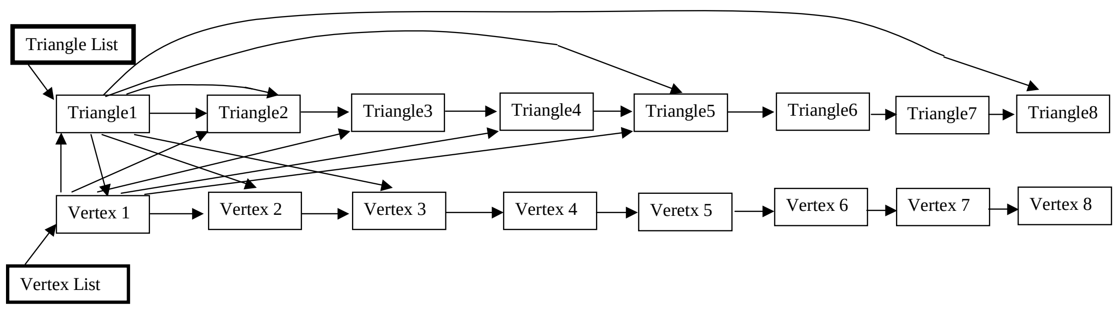


Figure 2.4 Data structure with topological information [21]

2.3 Dexel Model Structure Modification

Dexel models belongs to the category of decomposition models, which represent a solid as a collection of simple objects from a fixed collection of primitive object types [21]. The primitive element of dexel model is line segment (Figure 2.5).

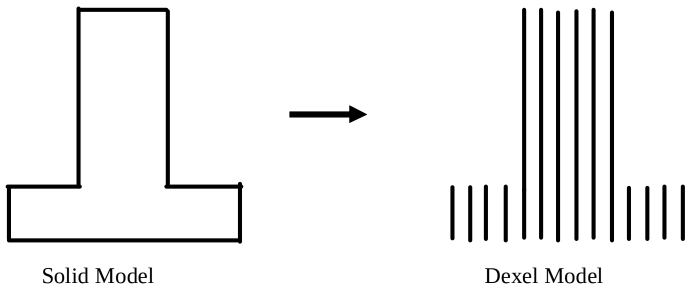


Figure 2.5 Dexel model [21]

In previous work by L. Cai [21], the dexel model was implemented (Figure 2.6). As we can see, dexel elements can be categorized into four categories 1) cover die dexel, 2) ejector die dexel, 3) part dexel, 4) undercut dexel. If triangles associated with each type of dexel can be identified, which die they belong to can easily be detected based on the dexel information. For example, if an ejector die dexel intersects with triangle A, triangle A is one of the triangles that constitute the interface between the part and the ejector die. Therefore, it can be used to construct the surface of ejector die's cavity. The same principle can be applied to triangles that intersect with the cover die and undercut dexels.

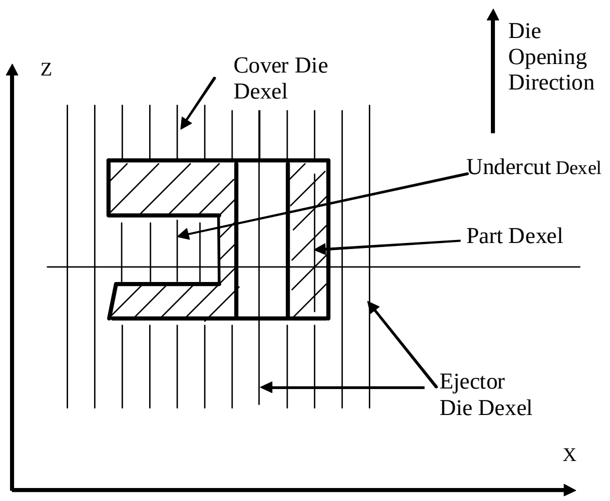


Figure 2.6 Part with through hole

The original dexel structure in CastView is sufficient for dexel-based modeling and reasoning. However, it provides no correlation information between each dexel element and the triangle it intersects with. In addition, it is incapable of identifying the difference between dexels going through a hole and dexels that are out side the part (Figure 2.6).

In order to establish and record the triangle-dexel correlation and to identify holes, the original dexel model is modified to provide additional information. Data structure details about the original dexel model and modified dexel model will be discussed in the next two sections consecutively.

2.3.1.1 Original Dexel Model Structure

The dexel model data structure used in this research was first introduced by L. Cai [21]. The concept was derived from z-buffer method which contains depth (z) value from a pixel to the nearest visible surface of the object in each element to refresh the screen [21].

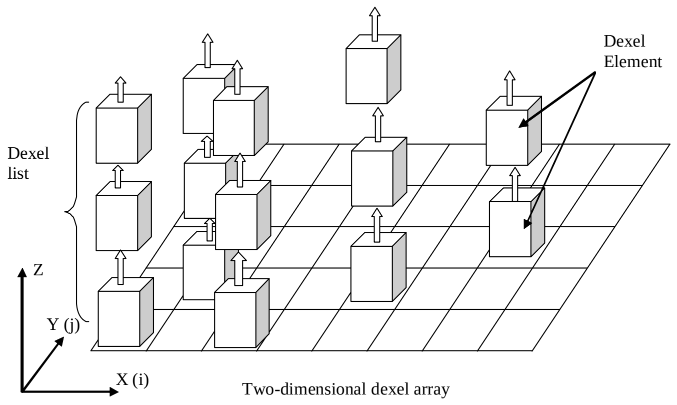


Figure 2.7 L. Cai's dexel model structure [21]

A two dimensional array (Figure 2.7) is used to record the location of each dexel list with coordinates (i, j) along the XOY plane. No matter what direction user defined as the die open direction, the STL model is rotated so that the die opening direction is along the Z-axis [21] (Figure 2.8), and the first dexel list will always start at (0, 0, 0). The z dimension of each dexel ray is from zero to the max z value of the part. The size of each dexel list depends on how many times the dexel ray stored in that list intersects with different triangles.

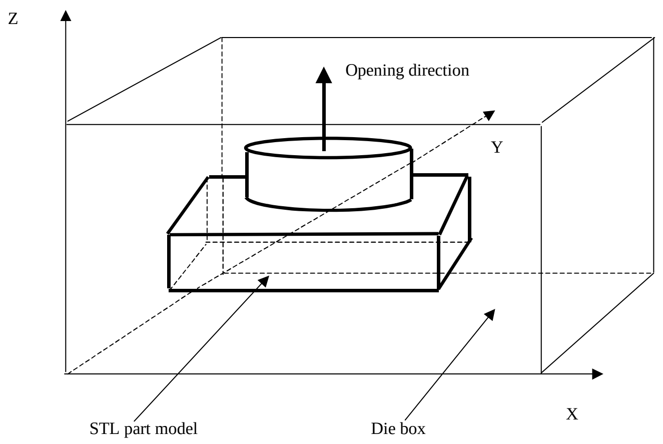


Figure 2.8 Part positioning in the coordinate system [21]

For each dexel element, it contains z coordinate of the intersection point, normal of the intersected triangle and type of the dexel (Figure 2.9). The z coordinate is the starting point of each dexel element and the ending point for its previous dexel element, excluding the first element in a dexel list. The normal is recorded for dexel type classification.

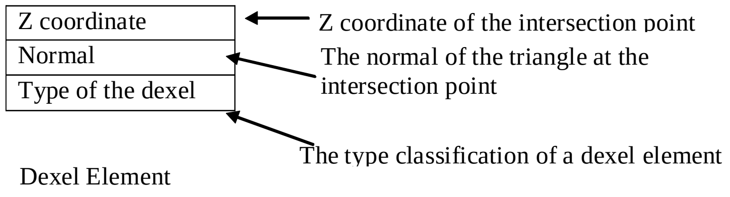

Z coordinate
Normal
Type of the dexel
Z coordinate of the intersection point
The normal of the triangle at the intersection point
The type classification of a dexel element
Dexel Element

Figure 2.9 A dexel element structure [21]

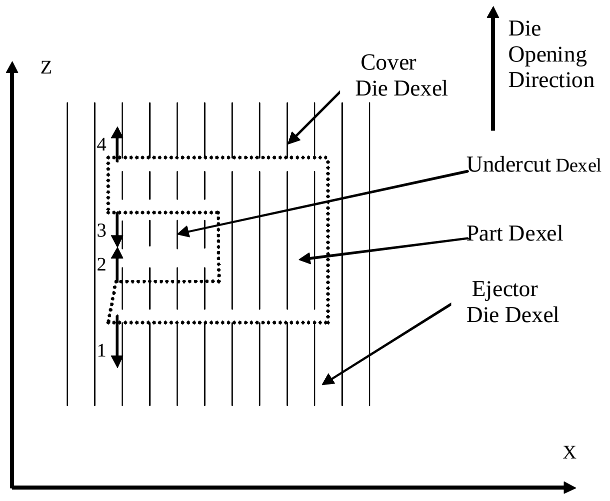


Figure 2.10 Four types of dexel

There are four types of dexel elements in original dexel structure (Figure 2.10): 1) part dexel, 2) undercut dexel, 3) ejector die dexel, 4) cover die dexel. A dexel that does not intersect with part is also called die dexel before it is divided into two pieces by the parting plane. The first dexel element with a normal (0, 0, -1) in a dexel list is defined as a ejector die dexel. The last dexel element with a normal (0, 0, 1) in a dexel list is called cover die dexel. If the z coordinate of a dexel element is positive and its next neighbor has a normal with a negative z value, then this element is classified as an undercut dexel. On the other hand, if the z coordinate of a dexel element is negative and its next neighbor has a normal with a positive z value, then this element is classified as a part dexel. By grouping different types of dexel elements, users can easily construct dexel models for part, die and even undercut. One disadvantage of this method is that due to its uniform structure, the accuracy of modeling largely depends on the pre-defined dexel resolution. If the resolution is too small, the model shape will have large tolerance comparing to original STL model.

2.3.1.2 Modified dexel model structure

As mentioned in the introduction, in order to obtain the correlation information and to identify holes using dexels, a modified dexel model structure is used in this research.

Besides the three member variables in a dexel element, an additional member is introduced into the structure—the index of the triangle with which the dexel element intersects. Since all the triangles are stored in a triangle face list, an index pointing to a specific triangle face can quickly guide the search algorithm to the correct position (Figure 2.11).

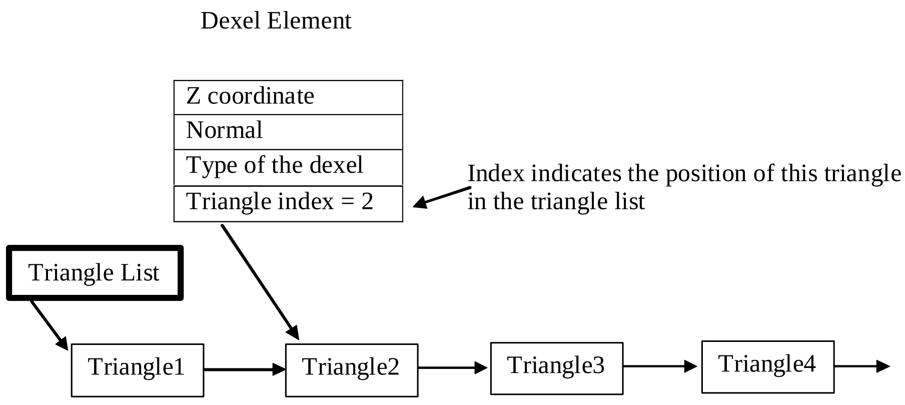


Figure 2.11 Modified Dexel structure

In addition to the four types of dexels, one more type, which is called the hole dexel (Figure 2.12), is added. To distinguish the difference between the hole dexels and the die dexels, certain computation and grouping algorithm have to be implemented through the whole dexel array. But from definition point of view, a hole dexel is characterized as a ray not intersecting with the part but located within the part.

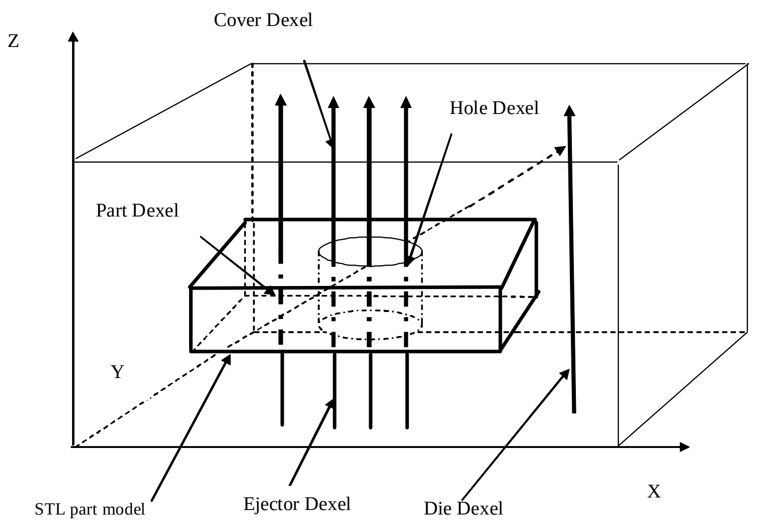


Figure 2.12 Redefined Dexel type

2.3.1.3 Definition of dexel related terms

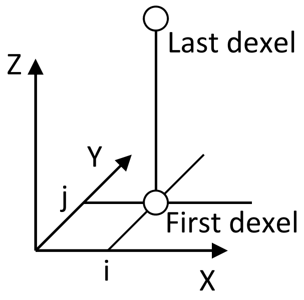
In this report, there are multiple places that different dexel terms are mentioned. In order to make the description clear in later chapters, the followings are the definitions of the terms:

- Dexel element: it is the basic segment of the dexel model structure. It contains a z value, normal, dexel type and triangle index.
- Dexel list: it is a linked list that contains all the dexel elements which have the same X and Y coordinates, (i, j). The order of the list is based on the z value of each dexel element.
- Dexel array: it is the two dimensional (on XOY plane) array that was displayed in Fig. 2.7. Each element of the array contains a dexel list.
- Dexel: in this thesis, dexel means the line segment composed by two known points. The coordinates of the starting point of this line is the targeted dexel element's coordinates (i, j, z) and the value of the ending point is the dexel element's next neighbor's coordinates (i, j, z').

2.4 Modeling Methodology

2.4.1 Dexelization Algorithm

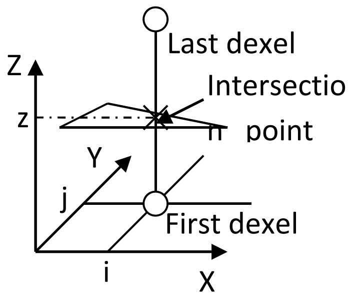
First, the STL model is oriented so that the open direction is the Z direction. Second, the dimension of the dexel array is calculated by interpreting the mathematical relationship between user defined dexel resolution and the actual dimension of the STL model. The result of the calculation will determine the size of the dexel array along x (i) axis and y (j) axis. Then each dexel list is initialized with two dexel elements (Figure 2.13).



First Dexel Element	Last Dexel Element
Z=0	Z= max Z
Normal=(0, 0, -1) (default)	Normal = (0, 0, 1) (default)
Type = Die Dexel	Type =Die Dexel
Tri_index = -1	Tri _index =-1

Figure 2.13 Initiation of dexel List

Finally, every triangle, which has a non-horizontal normal, is called out of the triangle list to intersect with each dexel ray. If a qualified intersection point is found, a new dexel element will be stored in the dexel list (Fig. 3.2). For example, dexel ray located at (i, j) intersected with the 3rd triangle in the triangle list, which has a normal of (0, 0, 1). The new dexel element will be inserted into the dexel list at location (i, j) of the 2 dimensional dexel list array, and its value is shown in Figure 2.14.



First dexel	Inserted dexel	Last dexel
Z=0	Z=z	Z= max Z
Normal=(0, 0, - 1)	Normal=(0, 0, 1)	Normal = (0, 0, 1)
Type = Die Dexel	Type(undefined)	Type =Die Dexel
Tri_index = -1	Tri_index=3	Tri _index =-1

Figure 2.14 Dexel list after dexelization

2.4.2 Classification of Dexels

At this point, all the dexel elements have been calculated and stored, but the type of each dexel is not yet defined. To classify die, part and undercut, we just need to compare the normal with its neighbor or check its location in a dexel list (Figure 2.15). The algorithm for determining hole dexels' locations will be discussed for the rest of this section.

```
For each dexel list in the dexel array:
    • If the list has exact two elements:
        1. Assign the type of first element as EJECTOR DIE DEXEL.
        2. Assign the type of the last element as DIE DEXEL.
    • If the list has more than three elements:
        1. Assign the type of the first element as EJECTOR DIE DEXEL.
        2. Assign the type of the last element as DIE DEXEL.
        3. Assign the type of the element just before the last element as COVER
           DIE DEXEL.
        4. For remaining elements:
            • If the z component of the normal of the element is the positive
              and the z component of the normal of the next element is
              negative, assign the type of the element as UNDERCUT
              DEXEL.
            • If the z component of the normal of the element is negative
              and the z component of the normal of the next element is
              positive, assign the type of the element as PART DEXEL.
```

Figure 2.15 Classification algorithm [21]

There has to be dexel type transformation when a hole exists (Figure 2.16). For a part without undercuts, the classification is very simple. The sizes of the dexel list can only be either 4 or 2. If the size of a dexel list is 4, this dexel must have intersected with the part. If its neighbor dexel list's size is 2, there is a type change. One possibility is that the dexel changes from part to die, the other possibility is that part dexel changes to a hole. To differentiate the two situations, a search algorithm is used. First, the whole dexel array is searched line by line, starting from the beginning of each line to the end. Suppose there have been no changes until the dexel list at (i, j). Dexel (i, j) has a size of 4, but its neighbor, dexel list (i+1, j), has a size of two and list (i+1, j) is not the last list on line j. Then dexel list (i, j) will be marked as a possible hole dexel, named P_DEXEL (i, j). Searching continues along the direction of i to the last segment of row j.

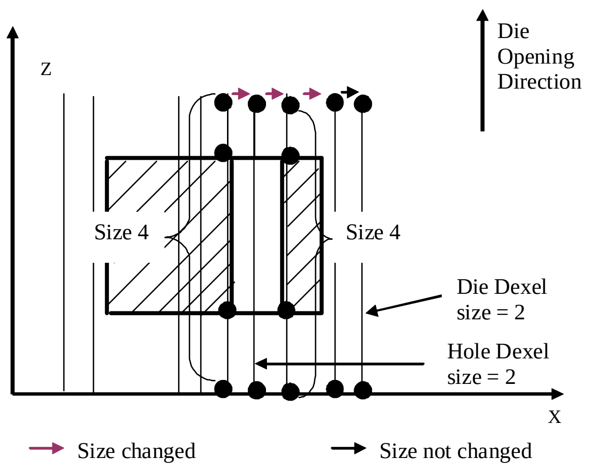


Figure 2.16 Size change difference between Hole Dexel and Die Dexel

If during the searching process, the size of dexel list (i, j+x) is 2 but the neighbor dexel list (i, j+x+1) has 4 dexel elements, stop searching and change the type of dexel elements of each list that located between (i, j) and (i, j+x) to hole dexel. After redefining the dexel type, repeat the same searching method for the rest of the dexel array.

For a complex shape with undercuts, an extra comparison algorithm must be applied prior the searching process. As show in Figure 2.17(a), a part, which does not have an undercut or a hole, can have 4 dexel elements in that dexel list. Every additional undercut will increase the dexel list size by two (Figure 2.17 (b)). Therefore, the total size of such a part is 4+2n, where n is the number of undercuts. A similar statistic can apply to any shape with holes. Every hole results in a size reduction by two (Figure 2.17 (c)). Therefore the total size will be 4+2n-2m (m is the number of holes). For any dexel list that has a size larger than 4, in order to accurately identify the type change, each dexel element in a list should be compared with its immediate neighbor. This comparison including type checking and z value comparison. For example, dexel element (i, j, k) is the kth part dexel in dexel list (i, j). Its z value is Z_1 and its next dexel element's Z value is Z_2. Suppose there is a virtual vertical line constructed by these two values, compare this line with every line composed by each non part dexel element's Z value, Z_3 of its neighbor dexel list (i+1, j) and Z_3's next Z value Z_4. If there is over lapping between the two lines (colored line showing in Figure 2.17), it indicates that there is a potential type change. The searching algorithm can take over the rest of the calculation at this point. Repeat the comparison and searching steps until all the dexel lists have been checked.

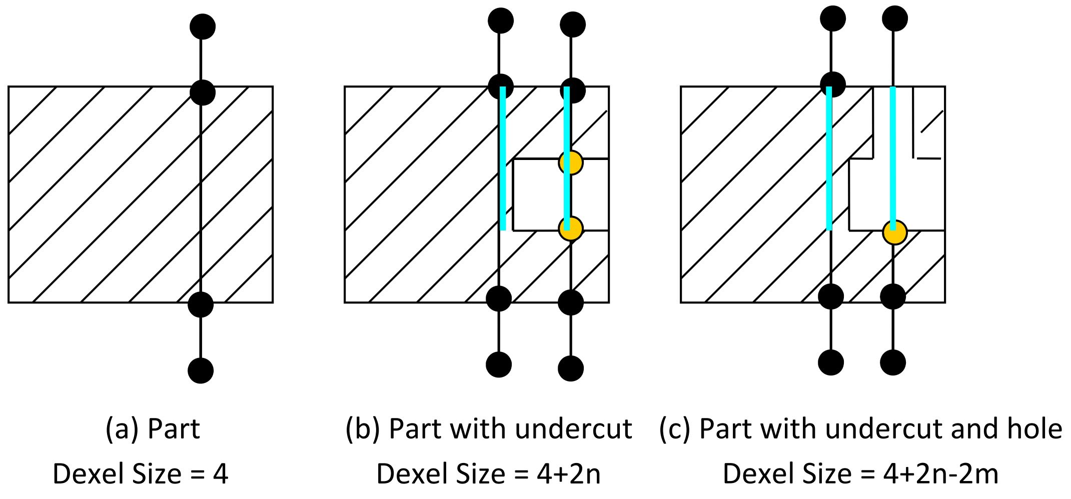


Figure 2.17 Size calculation for complex shape

Because the search algorithm only moves along the X (i) axis, if there is open space along the y axis, the die dexel will be misclassified as a hole dexel (Figure 2.18).

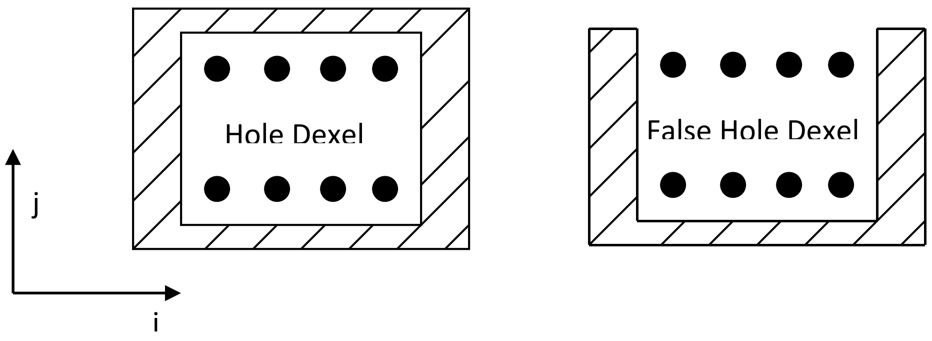


Figure 2.18 False hole dexel

Additional steps are needed to eliminate such errors. A similar search approach as described above is used here and this step should be performed after hole dexels are grouped (hole dexel grouping algorithm will be discussed in next section). The difference is, instead of searching along the x-axis, this time the search moves along Y direction. Suppose the first hole dexel of group 1 at location (i, j) is reached. First check its previous neighbor on Y direction, which is dexel list (i, j-1). If dexel (i, j-1) is not a part dexel, searching stops and the whole group was misclassified. All the dexels' type in this group should be changed to their original types. Otherwise keep searching all the dexel lists with location coordinates of (i, J) ($j+1 \leq J \leq \max(j)$). If before reaching the dexel list at (i, max (j)), dexel type changes to part dexel, searching should be stopped and move on to next row of the dexel array. If there is no type change after checking dexel element at (i, max(j)), dexel elements in this group were misclassified. Repeat this checking operation until all the groups are examined.

2.4.3 Grouping Hole Dexels

The result of previous dexel classification steps is that all the detected hole dexels are redefined. But at this stage, every dexel element is an individual. No information about location relationship is recorded in the system. In order to solve the misclassification problem mentioned in previous section and speed up the searching process, all the hole dexels should be grouped. The grouping method is based on the fact that only dexels which share a continuous space are in the same group.

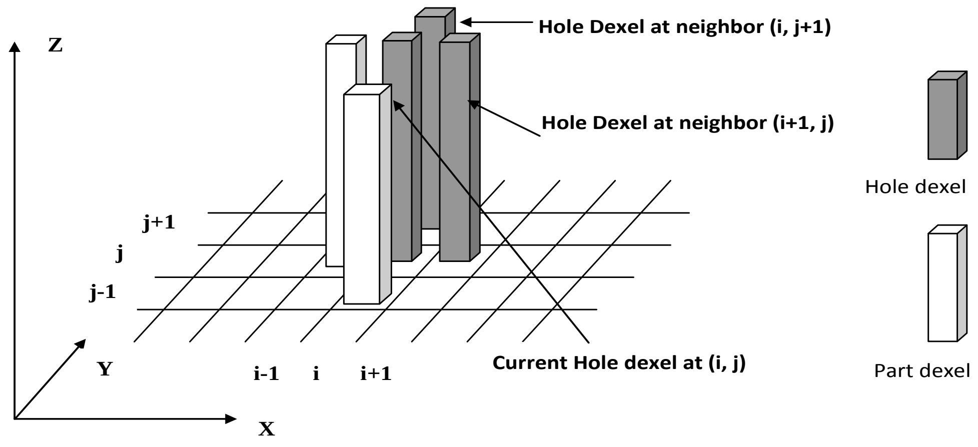

Z
Hole Dexel at neighbor (i, j+1)
Hole Dexel at neighbor (i+1, j)
Hole dexel
j+1
j
j-1
Y
i-1
i
i+1
Current Hole dexel at (i, j)
Part dexel
X

Figure 2.19 Grouping hole dexel

First, search through the dexel lists until find the first hole dexel is found in dexel list (i, j). This dexel element becomes the primary target and its data are pushed onto a stack. Then the first hole dexel is popped off the stack and its four neighbors, which are dexel list (i+1, j), (i, j+1), (i-1, j) and (i, j-1) are searched. If the line constructed by concurrent dexel's z value and its next z value over laps with its neighbor hole dexel, they belong to the same group (Figure 2.19). The neighbor hole dexel is pushed into the stack and comparison keeps repeating until the stack is empty. By the time the stack is empty, the hole dexel elements in the same group have been sorted out and marked with their group number. Same process is repeated until all the hole elements are checked and reclassified. The grouping algorithm is displayed in Figure 2.20.

```
i=0, j = 0;
While ( i < max number, j < max number)
    Get dexel list at (i, j)
        Current dexel = first dexel in dexel list at (i, j)
        While (current dexel != end of the list)
            If (current dexel is ungrouped hole dexel)
                Push current dexel in stack
                    While (stack is not empty)
                        Current dexel = pop the stack;
                        Assign current dexel the group number;
                        Find the neighboring hole dexels;
                        Push all the found neighboring undercut
                        dexels in stack;
```

Figure 2.20 Algorithm of grouping hole dexels

2.4.4 Grouping Undercut Dexels

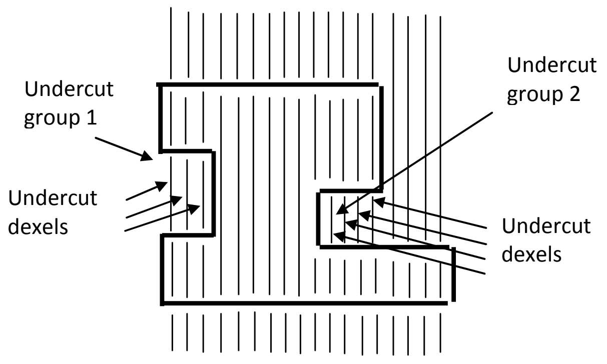


Figure 2.21 Individual undercut dexels and undercut groups [21]

Without grouping undercut dexels, each undercut dexel is just individuals and not related to its neighbor. Since one of the ultimate goals is to recognize the geometry of each undercut, grouped undercut dexels will be very helpful for creating and grouping undercut surface triangles which belong to the same undercut.

A grouping method similar to grouping hole dexels is utilized here. Once an undercut dexel in a list is identified, its four neighbors are checked to see if they have undercut dexel that overlaps with the first detected undercut dexel. If the answer is yes, then they are in the same group and the group number is assigned to each group's dexels (Figure 2.22). Repeat this procedure until all the individual undercut dexels are grouped.

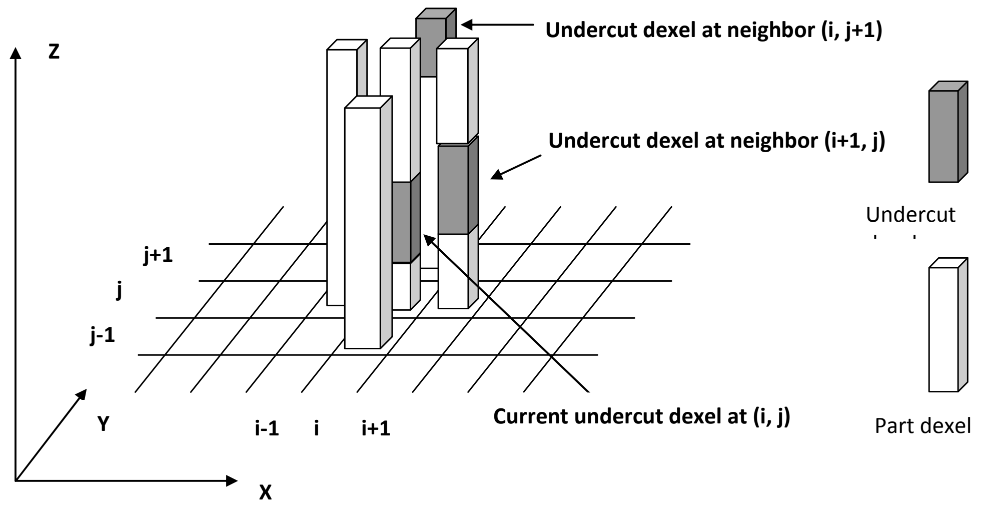

Undercut dexel at neighbor (i, j+1)
Z
Undercut dexel at neighbor (i+1, j)
Undercut
j+1
j
j-1
Y
i-1
i
i+1
Current undercut dexel at (i, j)
Part dexel
X

Figure 2.22 Find the neighboring undercut dexels [21]

2.4.5 Classification of Triangles

In section 3.2, a modified dexel structure was introduced. The reason driving this data modification is to associate an STL triangle with its dexel element and reveal its correct relationship with the cover die and the ejector die. All triangles intersected with die dexels can be classified based on their associated dexels types. However, triangles, which are parallel to the Z axis, have no dexel information, so they have to be classified by using a dexel type change checking method and the parting plane.

2.4.5.1 Classification of Dexel Intersected Triangle

Since during casting process, the shape of part is formed by the shapes of both die cavities, from geometric point of view, they both have the same surface triangles with normals pointing in opposite directions. If triangles of a part surface can be classified using the following two categories, 1) surface triangles that touch the cover die, so called cover die triangles, 2) triangles that are adjacent to the ejector die, named ejector die triangles, both cover die's and ejector die's cavity surface can be constructed by grouping each type of surface triangles into either cover die model or ejector die model.

To categorize triangles, the intuitive way is to compare the locations of triangles with parting plane's location. If a part is split in halves by the parting plane, triangles above the parting plane are classified as cover die triangles and triangles below the parting plane belong to the ejector die. However, this criterion is not sufficient. For example, as shown in Figure 2.23, triangles on the non- ejectable surface are classified as cover die triangles. The part cannot be ejected out of the ejector die in this configuration. Therefore, it is not sufficient to determine the relationship between triangles and dies just based on parting surface.

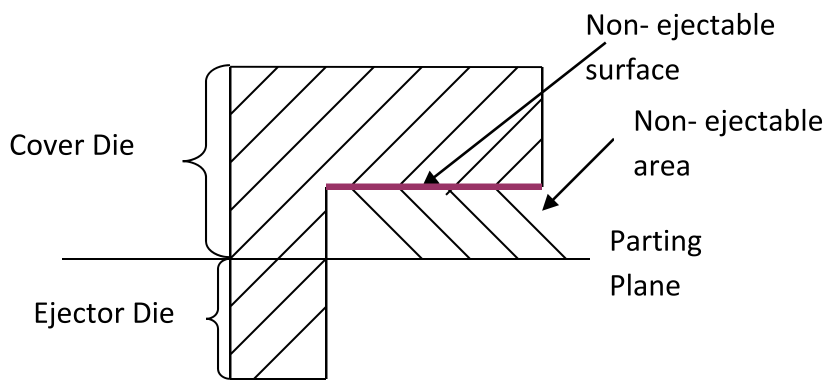


Figure 2.23 Non-ejectable die design based on parting plane

Triangles on a part's surface, that intersect with the cover die dexels, have the same shape and vertices as a triangular facet on the surface of the cover die. The only difference is that the two associated triangles have opposite normals. Therefore, if all the triangles intersecting with the cover die can be retrieved, the cover die cavity's surface can be reconstructed by connecting all the qualified triangles together. The same principle also applies to the ejector die. Whenever a triangle intersects with an ejector die dexel, by reversing its normal, a triangle on the ejector die cavity surface is identified. This principle stands true no matter where the parting plane is located (Figure 2.24). By recalling the triangle index stored in each dexel element, it is very easy to track all the triangles that intersect with die dexels. Based on this classification criterion, misclassified triangles as shown in Figure 2.23 are grouped in the correct category (Figure 2.25).

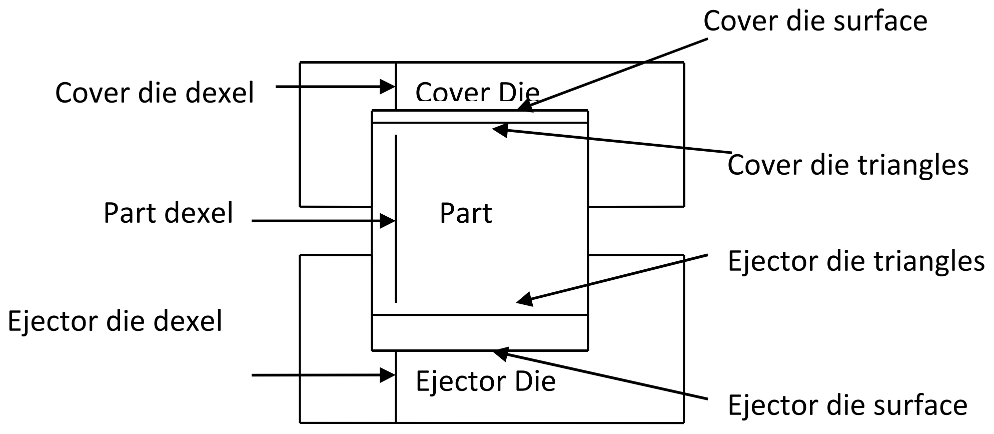


Figure 2.24 Correlation between triangles on part and die cavity surface

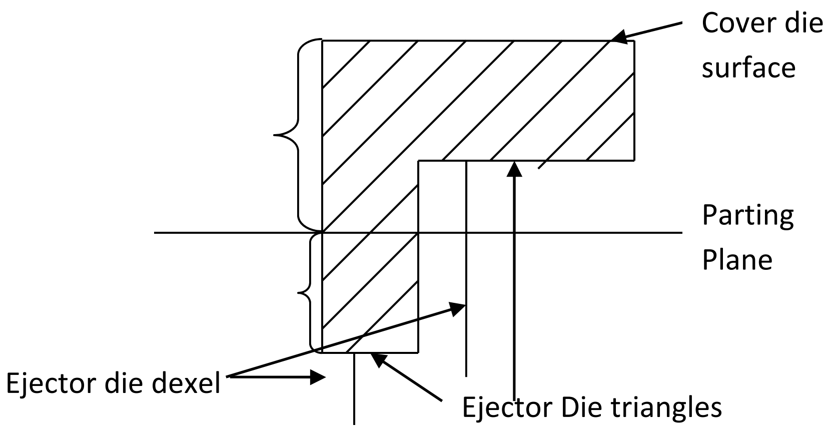


Figure 2.25 Triangle classification based on dexel information

2.4.5.2 Classification of vertical triangles

All triangles with a horizontal normal are parallel to the dexels. Therefore, there is no dexel data structure for this kind of triangles. Vertical triangles are divided into two sub-groups. The first group consists of triangles that should not be split or retriangulated by the parting plane no matter where the parting plane is located. This group is called non-split triangles. The second group contains triangles that intersect with the parting plane. They should be sliced along the parting surface and are called split-triangles (Figure 2.26).

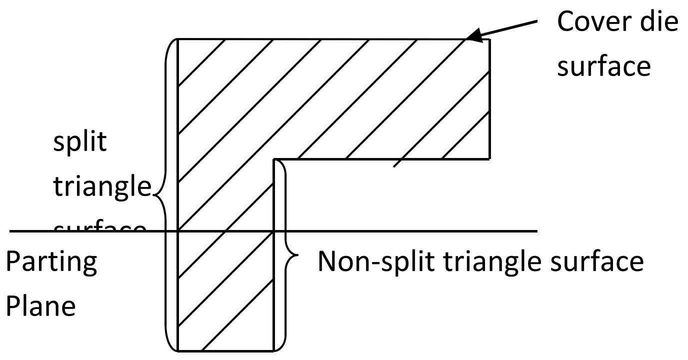


Figure 2.26 Two sub-group of vertical triangles

To separate the two groups of triangles, a new classification approach is used. After the initial classification, triangles belonging to the cover die are marked as cover die triangles and triangles of the ejector die are defined as ejector die triangles. Both of their check flags are set to be “true”. Only vertical triangles are left undefined. Search starts from the beginning of triangle list and the following calculation is performed only when the normal of triangle is (x, y, 0).

The targeted triangle is projected to the XoY plane as a line L. Assume the left end point of line L is the starting point, point1, and the right end point is point2. If the line is parallel to Y axis, then point 1 should be the lower ending point of line L and point 2 is the upper ending point. Suppose that the maximum x and y values of the line L are max_x and max_y and minimum x and y values are min_x and min_y.

There are four rules that can be used to make decisions about the triangle type. 1). if the dexel type changes between ejector die dexel and part dexel, then the triangle is an ejector die triangle. 2). if the dexel changes between cover die dexel and part dexel, the triangle should be classified as a cover die triangle. 3) if the type changes between dexel undercut and part dexel, the triangle is the side surface triangle of an undercut. 4) if the type changes between die dexel (dexel not intersectes with part) and part dexel, the triangle is sliceable and parting plane information is required to finally define its characteristics. Type changes between the hole dexels and part dexels will be discussed later in this chapter.

In order to get accurate classification results, every dexel element located nearest to the projected line should be checked to see if there is any dexel type change (Figure 2.27).

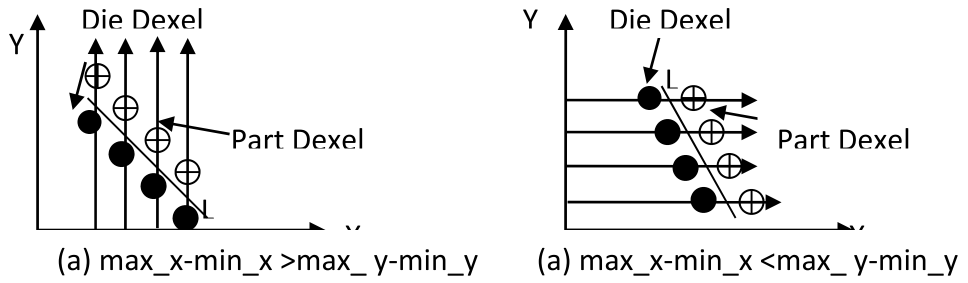


Figure 2.27 Dexel type check around the projected line

One critical step before comparing the dexels' types is to find the right neighbor dexels. First, the neighbor dexel searching algorithm starts from point1. Assume point 1 is the current testing point, the algorithm for finding its neighbor dexel is shown in Figure 2.28 and Figure 2.29. Second, all the other testing points on line L and their neighbor dexels should be calculated. There are two possible situations of defining testing points.

- If (max_x-min_x) is larger than (max_y-min_y), the testing points should be the intersection points of line i=i++ and line L (Fig. 3.10(a)). (i<=max_x/dexel size)
- If (max_x-min_x) is smaller or equals to (max_y-min_y), the testing points should be the intersection points of line j=j++ and line L (Figure 2.27(b)). (j<=max_y/dexel size) and line L (Figure 2.27(b)).

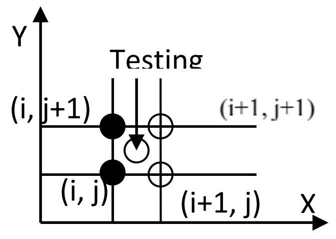


Figure 2.28 Four neighbor dexels of a testing point

- *Divide the x coordinate of point P1 by dexel size, the result is denoted as I;*
- *Divide the y coordinate of point P1 by dexel size, the result is denoted as J;*
- *If I and J are integers, find the smaller integer values of I and I, record them as i and j;*
    - *The left lower neighbor is dexel (i, j);*
    - *The left upper neighbor is dexel (i, j+1);*
    - *The right lower neighbor is dexel (i+1,j);*
    - *The right upper neighbor is dexel (i+1, j+1);*
- *If I is a integer, neighbor of the testing point is*

dexel(I-1, j) and dexel (I+1, j);

- *If J is a integar, neighbor of the testing point is*

dexel (i, J-1) and dexel (i, j+1)

Figure 2.29 Algorithm for finding four dexel neighbors of the starting point

For the first situation, suppose the intersection point is P with coordinates(x, y).

$$j = (\text{int})(y / dexel\ \ size)$$
$$i = x$$

So the intersection point's adjacent two neighbors as shown in Figure 2.27(a) are dexel (i, j) and dexel (i, j+1).

For the second situation,

$$i = (\text{int})(x / dexel\ \ size)$$
$$j = y \qquad ,$$

and its two neighbors should be dexel (i, j) and dexel (i+1, j).

After finding each intersection point's adjacent dexels, compare their types based on the four rules mentioned early in this section. It is easy to identify each vertical triangle's category.

Back to triangles that are sliceable. If the triangle does not intersect with parting plane and is above parting plane, it belongs to the cover die, otherwise it belongs to the ejector die. If it intersects with the parting plane, it will be split along the parting plane surface and will be retriangulated.

Figure 2.30shows the result of triangle classification. The light blue colored triangles are the cover die triangles and green colored triangles belong to the ejector die. Blue colored triangles are split triangles. Hole side surfaces, which intersect with the parting plane, are in orange.

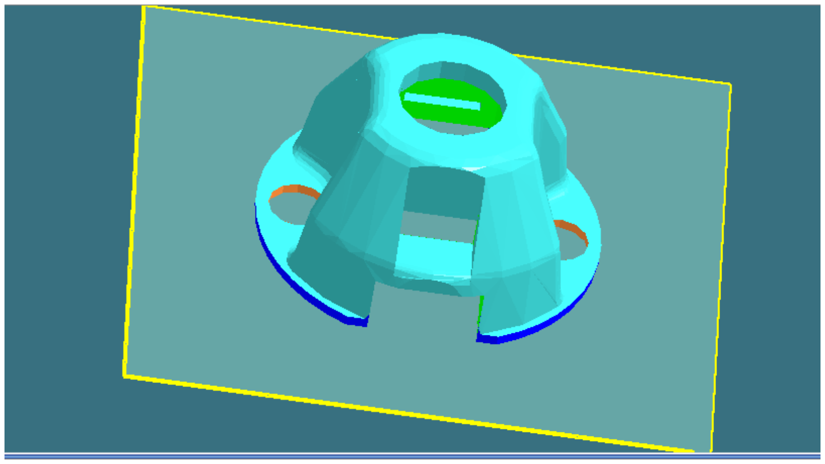


Figure 2.30 Triangle classification results

2.4.6 Generation of Hole Contour

Up until this point, all the triangles except triangles located in a through hole from the part geometry have been classified and reserved to construct the die model. However, there is still information missing. If there are holes on a part, there should be surfaces corresponding to the hole on both dies as shown in

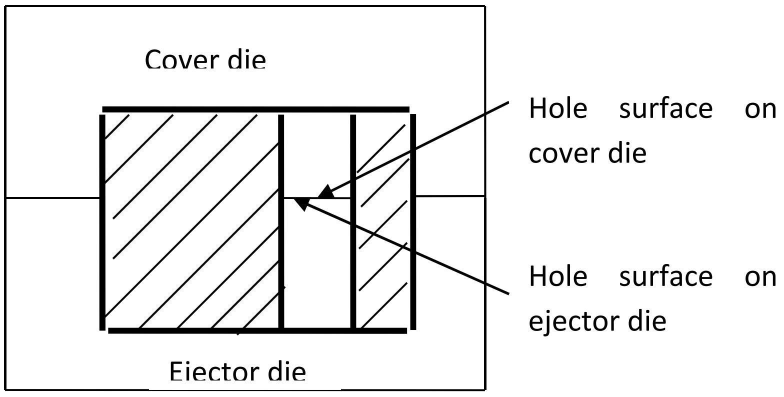


Figure 2.31. This information cannot be directly inherited from a part STL model, therefore an additional algorithm is required to specified this special feature and create new surfaces.

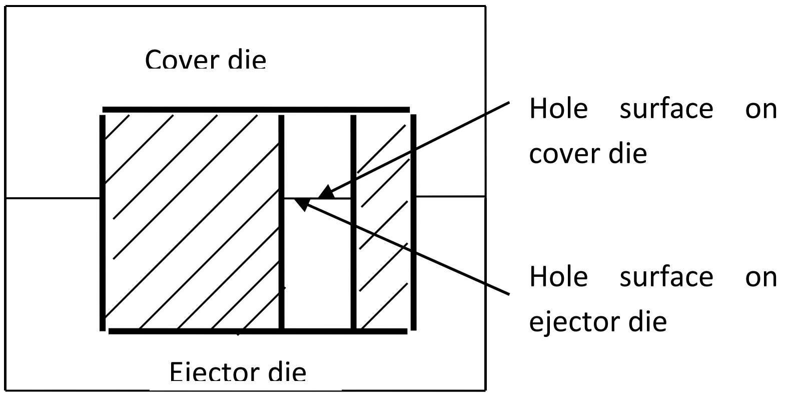


Figure 2.31 Constructed hole surface for cover die and ejector die

After grouping hole dexels, one more calculation is carried out at the same time as the grouping process to collect more useful information, such as the outer boundary of a hole dexel group and the hole contour. The group searching is along Y (j) axis. Whenever there is dexel type change between part dexel and hole dexel, the part dexel is marked as a potential hole outer boundary dexel. If at the end of the grouping process, all the hole dexels are accurately classified, the outer boundary dexel information is stored at the same time the grouped dexels are stored.

After the hole dexel grouping calculation, each triangle corresponding to the outer boundary dexel is picked out of the triangle list. By comparing its normal with its three neighbors' normals, whenever there is an angle between the two normals equal to 90 ± 3 degree (the 3 degree tolerance is arbitrary), the shared edge between those two triangles is one element of the hole contour. At this point, the hole contour element is inserted into a hole contour list with geometric information which will be useful for further analysis (Figure 2.32).

i value of the dexel list
j value of the dexel list
Triangle index
Starting point
End point

Figure 2.32 Data structure of a hole contour element

After all of the hole contour elements are inserted into one temporary hole contour list, they are resorted with geometry information and topology. Hole contours are displayed in yellow loops as shown in Fig 3.21.

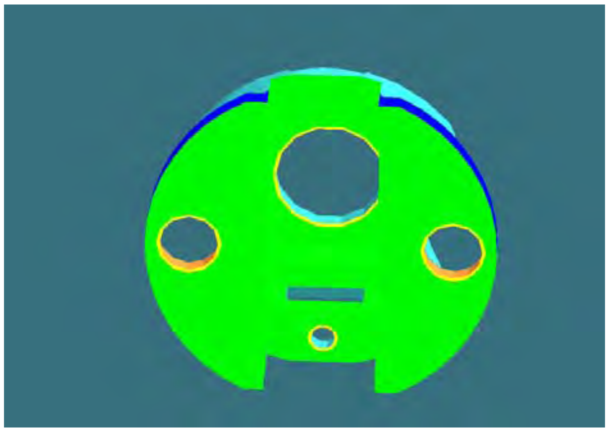


Figure 2.33 Hole contour

2.4.7 Generation of Undercut Surfaces

A geometry model of an undercut can be constructed by three portions of data. First ones are undercut top/bottom surfaces. Second ones are undercut side surfaces which can be retrieved from the part STL model, so called "type I undercut side surface" in later sections. The last category contains undercuts side surfaces which are unknown and need to be created (type II undercut side surface) (Figure 2.34).

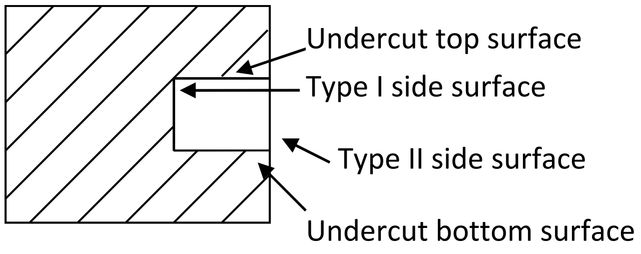


Figure 2.34 Part with undercut

Triangles belonging to undercut top and bottom surfaces can be located by using undercut dexels information. If there is an undercut dexel element in a dexel list, the triangle, with which the undercut dexel intersects, is one of the triangles that belong to undercut bottom surface. On the other hand, the triangle, which intersects with the next dexel element of the undercut element in the same dexel list, is one of the top surface triangles.

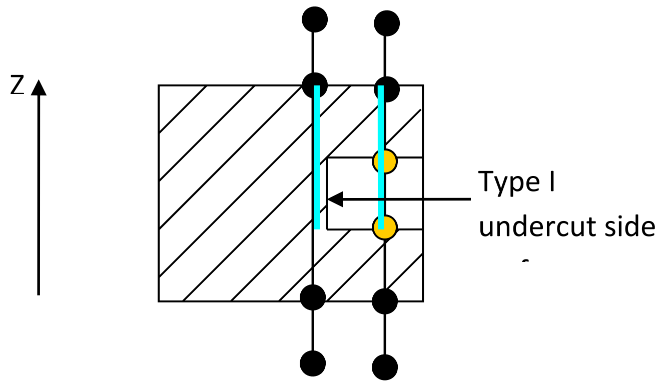


Figure 2.35 Type I undercut side surface identification

To build type I side surfaces of an undercut, the classification of vertical triangles method can be used (Figure 2.35). If there the dexel type changes between a part dexel and an undercut dexel, the triangles located between the two kinds of dexels are type I undercut side triangles.

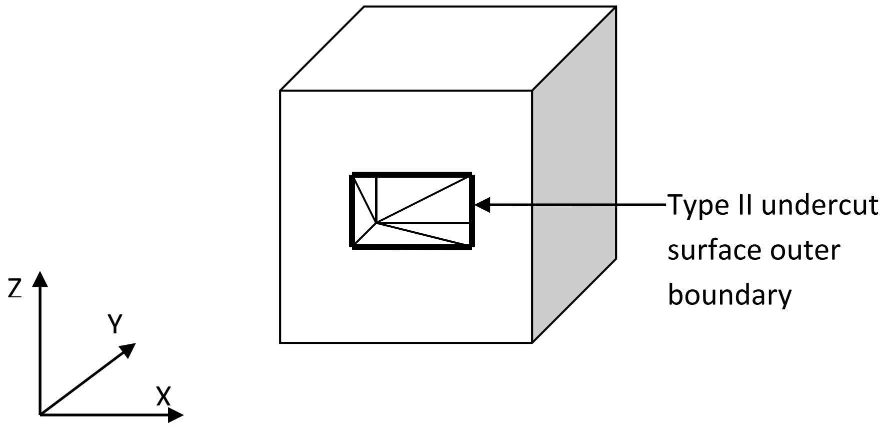


Figure 2.36 Type II undercut side surface with outer boundary

Because there are no existing triangles to construct the type II undercut side surface, the surface's outer boundary has to be generated (Figure 2.36), so that triangulation can be applied to create the surface. The boundary is a loop composed of edges, which are the common edges of undercut top/bottom triangles and cover/ejector triangles. By collecting all the common edges and sorting them in loops, the boundary can be built. A special data structured was created to record the boundary edges' information (Figure 2.37).

Starting point of edge
End point of edge
Undercut dexel group number
Triangle index

Figure 2.37 Data structure of undercut boundary edge

The algorithm for computing and sorting boundary edges is described in Figure 2.38. After all the outer boundary loops are sorted, triangulation should be applied to the loops to construct type II undercut side surfaces. Details about triangulation will be discussed in next chapter.

```
For each dexel top/bottom triangle:
        •  Find 3 neighbor triangles:
              For(i=0;i<3;i++)
              {
              If(neighbor[i] is cover die triangle or ejector die triangle)
              The common edge of the neighbor and undercut triangle is on the Type
              II undercut side surface boundary and is stored in a temporary
              boundary edge list L[i](i=0).
              }
After all the edges of out boundary are stored in the edge list:
              1.  start a new boundary edge list L[i++];
              2.  move the first edge from the temporary list to the new list as edge1.
              3.  for(edge2=first element of L[0];edge!=last element of L[0];
                  edge2++)
                  if( ending point of edge1 = start or ending point of edge2)
                  move edge2 to list L[i++];
              4.  repeat step 3 until all the edges in L[0] are sorted.
```

Figure 2.38 Calculation and sorting algorithm of undercut outer boundary loops

2.4.8 Grouping Undercut Surfaces

At this point, each undercut surface is created, but the connection between each surface is still missing. To build accurate undercut model, surface-grouping method is used.

Suppose there are n groups of undercut dexels from the grouping results. Undercut top/bottom triangles that intersect with group i^{th} undercut dexels are in i^{th} group too. As for type I undercut side surface triangles, if they are located between i^{th} undercut dexels and part dexels, they belong to group i. Since all the type II surfaces' boundary edges contain dexel group number, by importing the group number into triangles generated from the boundary edges, they can easily be grouped.

2.5 Triangulation and Retriangulation

2.5.1 Introduction

Triangulation is a method that creates a group of triangles from a series of ordered vertices (Figure 2.39).

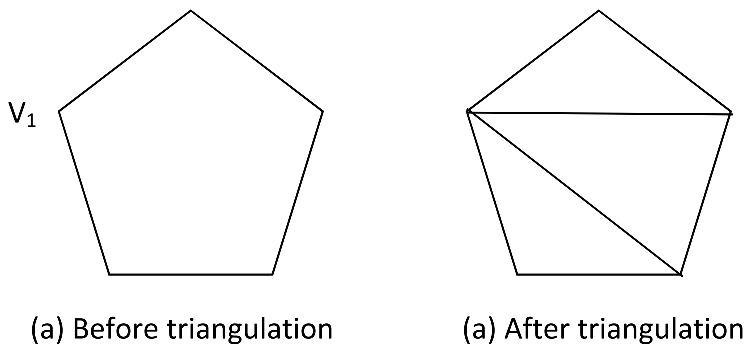


Figure 2.39 Comparison of geometry shape before and after triangulation

Whenever there are holes on a part, there should be new surfaces generated corresponding to the holes on the die STL model. In addition, in order to create accurate undercuts models, type II side surfaces of undercuts require creating new surfaces. One way to construct these surfaces is using a triangulation method.

Besides constructing new triangular surfaces, there are situations that retriangulation is needed. Retriangulation means rebuilding triangles from existing triangles corresponding to feature changes, e.g. triangle split by parting plane (Figure 2.40).

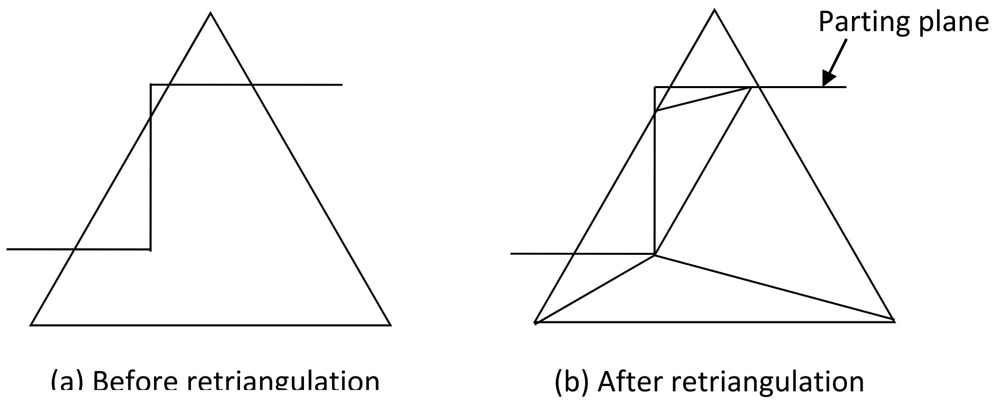


Figure 2.40 Comparison between original triangle and retriangulated triangles

2.5.2 Triangulation

Hole surfaces and undercut type II side surfaces are two of the major situations that require triangulation. Since their retriangulation method is the same after obtaining the contour information, they will be discussed in next section simultaneously. In addition, the interface information between cover die and ejector die is not contained in the part STL model, therefore it also needs to be triangulated. Discussion about die interface triangulation will be introduced in section 4.2.2.

2.5.2.1 Two triangulation methods

Two types of triangulation methods are used. For a convex contour loop, the triangulation algorithm, which will be called simple triangulation, is indicated in Figure 2.41. This method is easy to use and do not impose restrict requirements on how the contour should be ordered or what the data tolerance should be. Hence, in situation that contours are convex shaped, this method is applied.

However, when the shape of part is complicated, the first triangulation method will get inaccurate results. So for concave shaped contours, a second triangulation method, called polygon triangulation, is utilized.

```
Temp_contour_element= the first element of the contour, C1:
    1. First vertex of triangle =first point of C1;
    2. Second vertex of triangle=second point of C1;
    3. Temp_contour_element=next element of the contour;
    4. Third vertex of triangle=second point of C2;
    5. calculate triangle normal using 3 vertices coordinates;
    6. if(triangle normal pointing to the same direction as parting plane)
        • add the new triangle to ejector die STL model;
        • add another new triangle  with same vertices and reversed normal to cover
          die STL model;
       else
        • add the new triangle to cover die STL model;
        • add another new triangle  with same vertices and reversed normal to ejector
          die STL model;
        •
    7. for (i=0;i<=size of contour-3;i++)
            • second vertex of triangle = third vertex of triangle;
```

Figure 2.41 Algorithm of triangulation hole surface

2.5.2.2 Hole or undercut surface triangulation

A hole surface is created by triangulating a hole contour. There are two types of hole contours that will be used in this triangulation method (Figure 2.42). The first one is calculated by identifying edges that surround grouped hole dexels. Each edge element in this type of contour is one of its triangle's edges. The second type of hole contour occurs only when parting planes intersect with holes. Since elements of the second type are part of the intersection line contours created by parting plane and the part, they are already computed and sorted when the die configuration is defined. But because not just hole contours are on the contour list (Figure 2.43), they have to be separated from others.

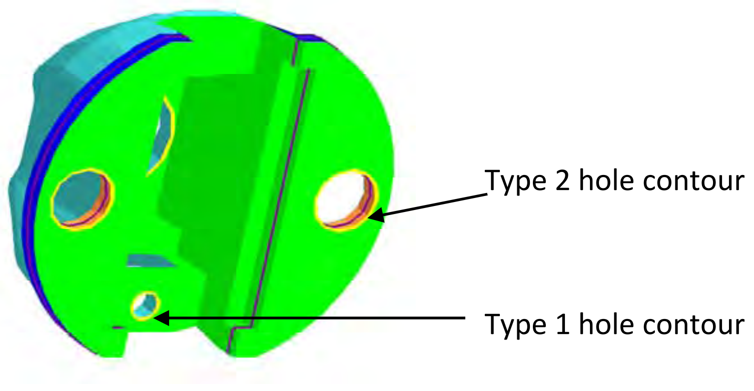


Figure 2.42 Two types of hole contours

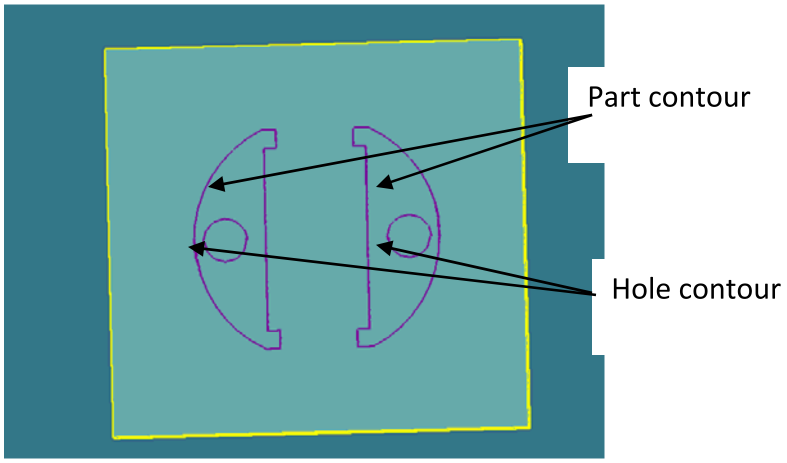
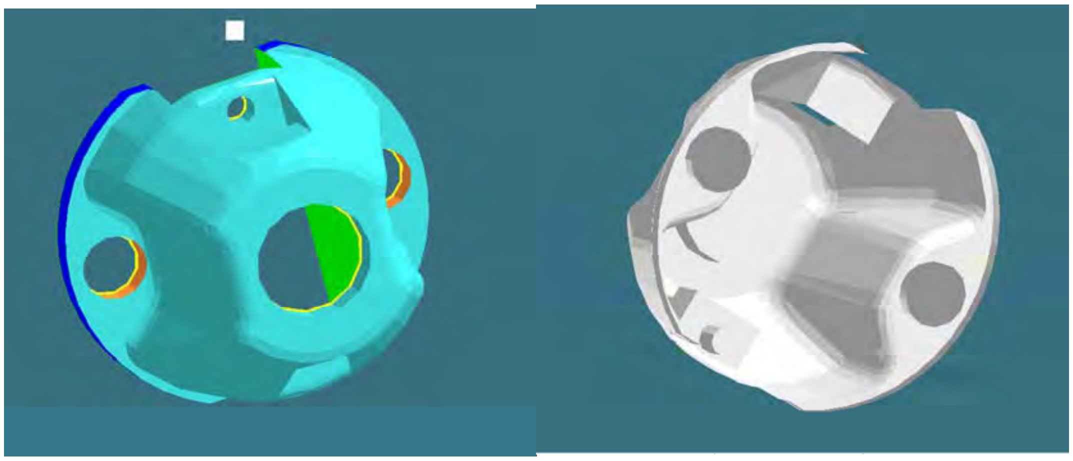


Figure 2.43 Contour types contained in die configuration contour list

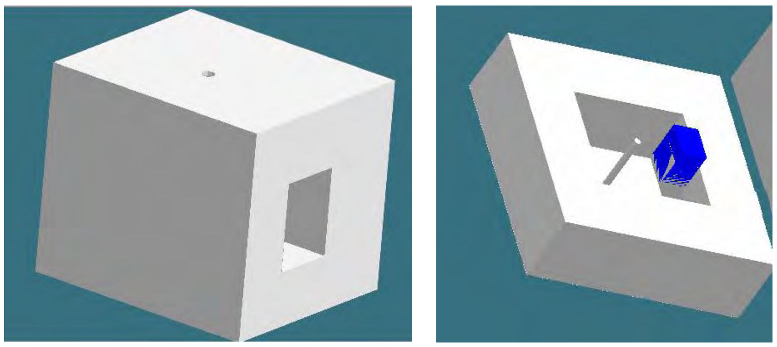
Since each contour element contains its associated triangle information, by checking its related triangle type, the contour type can be determined. If the type of the related triangle is a hole triangle, then the contour is a hole contour.

Knowing the contour, triangulation is carried out following the calculation steps as described in Figure 2.41. The same triangulation algorithm is used to create undercut type II side surface. One example of triangulated the hole surface is shown in Figure 2.44. One example of triangulated undercut surface is shown in Figure 2.45.



Part Cover die cavity

Figure 2.44 Part with hole and die cavity with triangulated hole surface



Part with undercut Die model with undercut model

Figure 2.45 Example of dexel model with constructed surfaces

2.5.2.3 Die interface triangulation with simple parting plane

Die interfaces (shaded area shown in Figure 2.46) are the faces where two die models mate with each other. To construct a complete die model, those faces needs to be triangulated.

For cases when the parting plane is a single plane, two types of contours are required for applying triangulation method. The first contour is a loop contains 4 edges created by the 4 intersection points of the parting plane and die box, which are ordered in counter clockwise direction. The second type of contour is composed of common edges that are shared by cover die triangles and ejector die triangles. The former type of contour is called outer contour, and the latter is called inner contour.

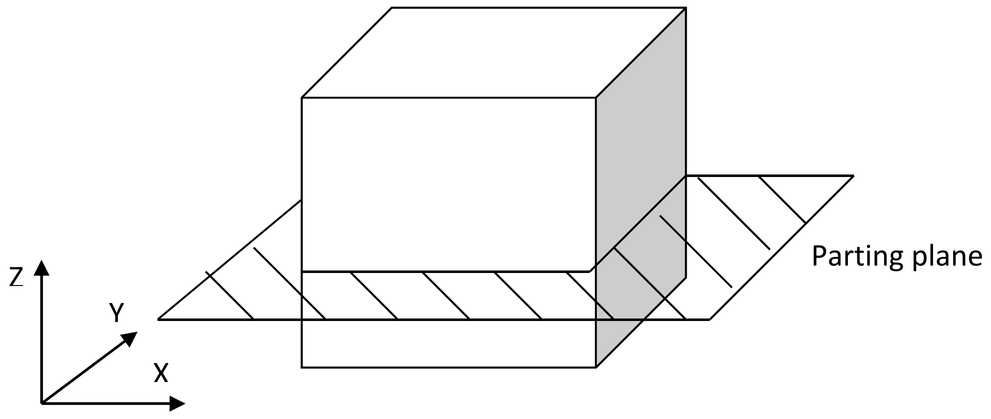


Figure 2.46 Cover die and ejector die interface

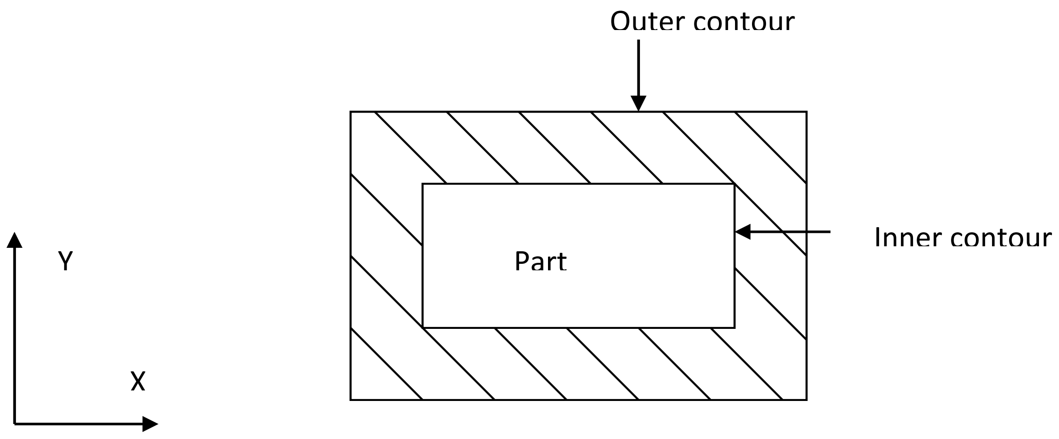


Figure 2.47 Triangulation area

The shaded area in Figure 2.47 is the area that should be triangulated to form a die interface. To construct the outer contour, the calculation is simple. Just intersect parting plane with die box and store the intersection points in the right order.

The second contour is more complex. The simplest case is that if the outer contour is part of the contour list generated when the die configuration is defined and contains edges that belong to sliceable triangles, then this contour is the inner contour (Figure 2.48).

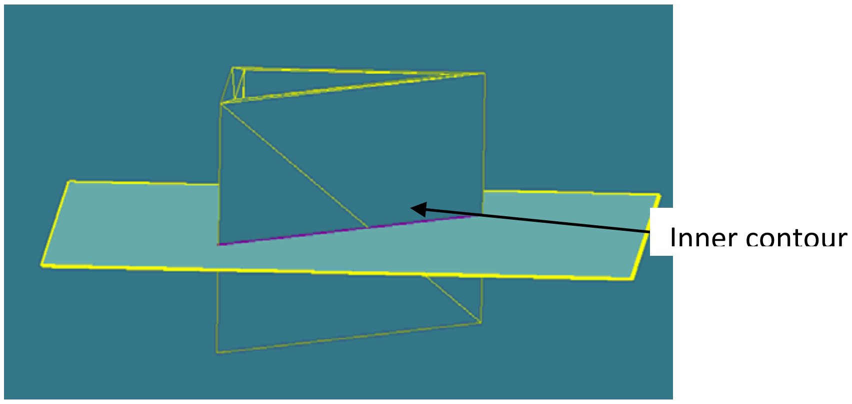


Figure 2.48 First method to find inner contour

The second situation can be derived from the situation mentioned above. If the inner contour also has member edges that belong to undercuts (Figure 2.49), for each group of undercuts that contribute edges to this inner loop, the contributed edges should be erased from the contour list and one additional edge should be added into the contour list as a substitute for the deleted edges (Figure 2.50).

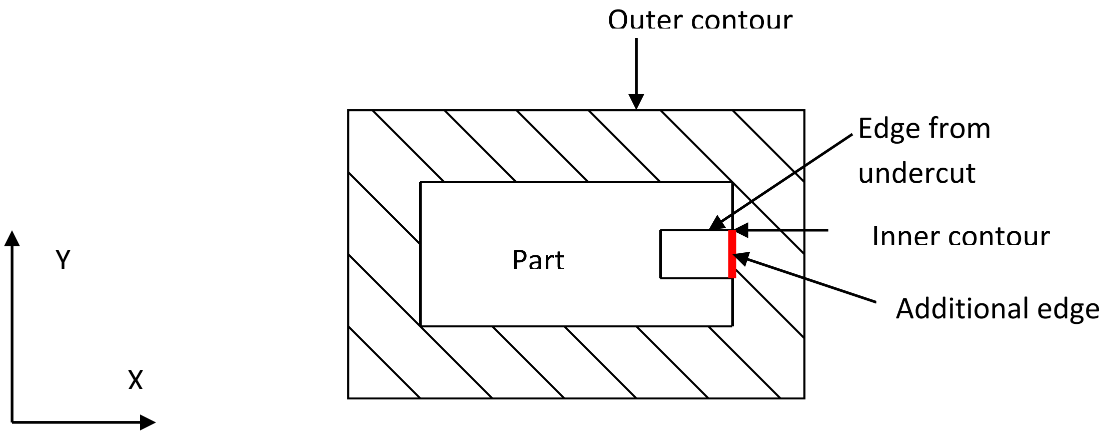


Figure 2.49 Display of inner contour with undercuts' edges

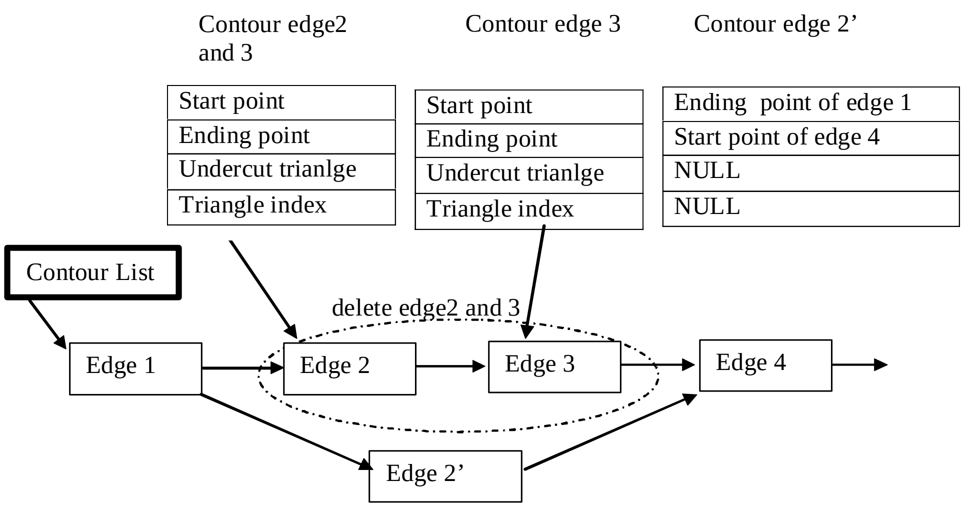


Figure 2.50 Data structure of inner contour with undercuts’ edges

The two ending points of the additional edge are the points which connect one edge from a sliceable triangle and an edge from undercut triangle.

The third situation is similar to the second. Instead of having inner contour going across undercuts, it intersects with holes (Figure 2.51).

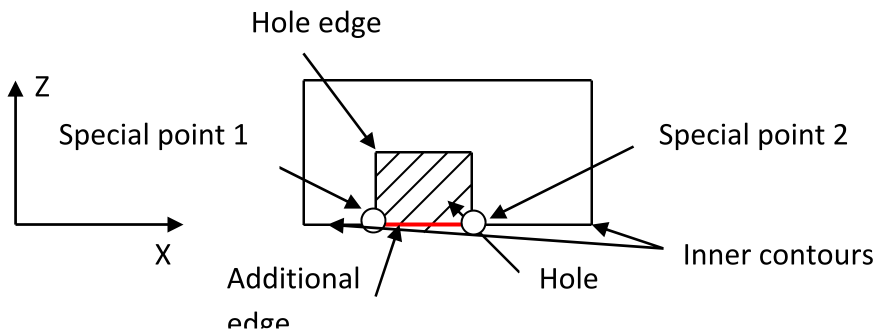


Figure 2.51 Inner contour intersects with holes

In this case, there are two inner contours. To combine the two loops together, two special points are identified first. If a point is shared by one inner contour edge and one hole edge, it is a special point. The additional edge contains two special points.

At this point, all the necessary loops have been structured. Because the loops might have complex shapes, polygon triangulation method is applied in the next step.

2.5.2.4 Die interface triangulation with stepped parting plane

If a stepped parting plane is constructed, polygon triangulation approach is used, but the contour structure is different. Instead of two types of contours, for each face of a stepped parting plane, only one contour is formed in a counter clockwise direction (Figure 2.52).

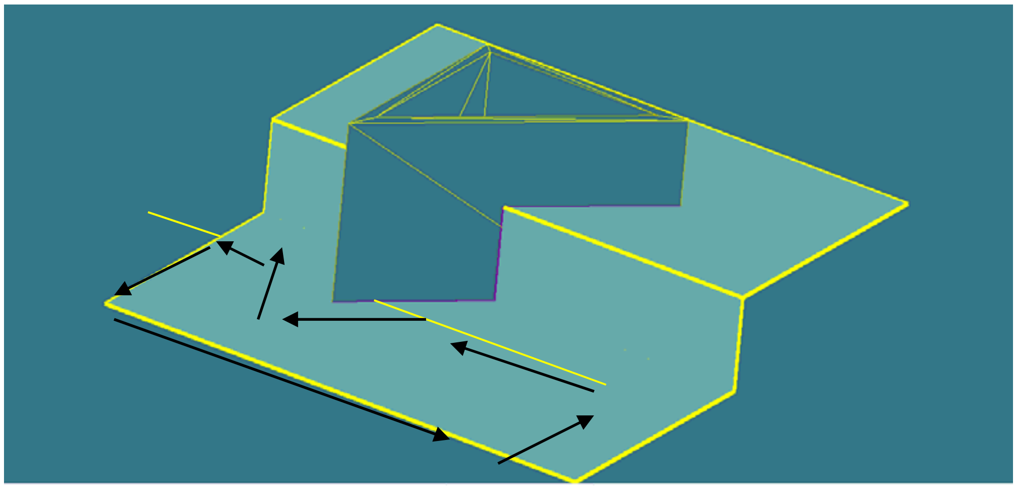


Figure 2.52 Structure of stepped parting plane contour for triangulation

The stepped parting plane problem is decomposed into a single plane problem. Consider one plane at a time. Find out all the contour edges that are located on each plane and then order them in counter clockwise direction. Detailed algorithm is introduced in Figure 2.53.

```
For (j=0; j<number of planes; j++)
    • Insert 4 edges of planes into a empty edge list;
    • For(i=0; i<size of contour; i++)
        o Edge[i]=i^th edge element of contour;
        o If(edge[i] is on plane j)
            ▪ Insert edge[i] into edge list;
        o If(one point of edge[i] P is on one of the 4 plane edges edge[n]
            ▪ Ending point of Edge[n]=P;
            ▪ Insert a new edge with starting point P and another point
              of edge[i] as the ending point;
    • Sort the loop in required order
```

Figure 2.53 Algorithm of creating stepped parting plane contours

2.5.3 Retriangulation

2.5.3.1 Retriangulation of sliceable triangle with single parting plane

When sliceable triangles intersect with a single parting plane, the triangle is cut into two pieces. One piece has vertices that are above parting plane and belong to the cover die and another piece belongs to ejector die (Figure 2.54). The algorithm for retriangulation is described in Figure 2.55.

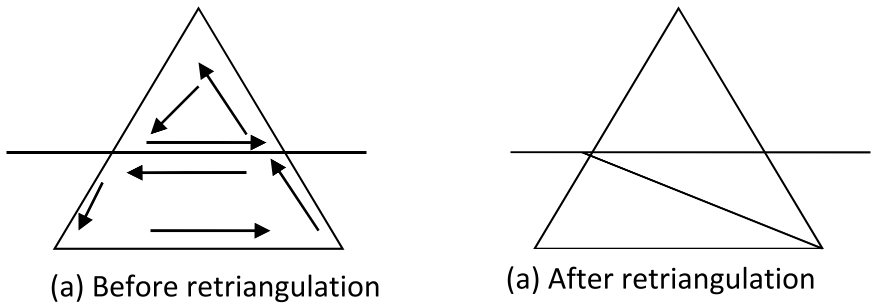


Figure 2.54 Retriangulation loops and results.

```
• If(three vertices are above or on the parting plane_
    o It is a cover die triangle
• Else If(three vertices below the parting plane)
    o It belongs to ejector die trianlge
• Else
    o p1= contour edge starting point
    o p2=coutour edge ending point
    o insert triangles three edges into a new edge list
    o insert contour edge into the new edge list
    o reverse contour edge, insert it too
    For(i=0; i<3; i++)
    {If (p1 or p2 are on triangle edge i)
    Ending point of triangle edge =p1 or p2;
    new edge starting point =p1 or p2;
    new edge ending point =ending point of triangle edge i;
    insert new edge;}
• sorting the edges into two loops as shown in Fig. 4.16.
• use simple triangulation method to triangulate any loops have more than 3 edges.
```

Figure 2.55 Algorithm for retriangulated sliceable triangle intersected by single parting plane

2.5.3.2 Retriangulation of sliceable triangle with stepped parting plane

Sliceable triangles, which intersect with multiple planes, are split into multiple and complex shaped loops (Figure 2.56). By separately triangulating each loop, the whole triangle is retriangulated. To triangulate irregular loops, polygon triangulation is implemented.

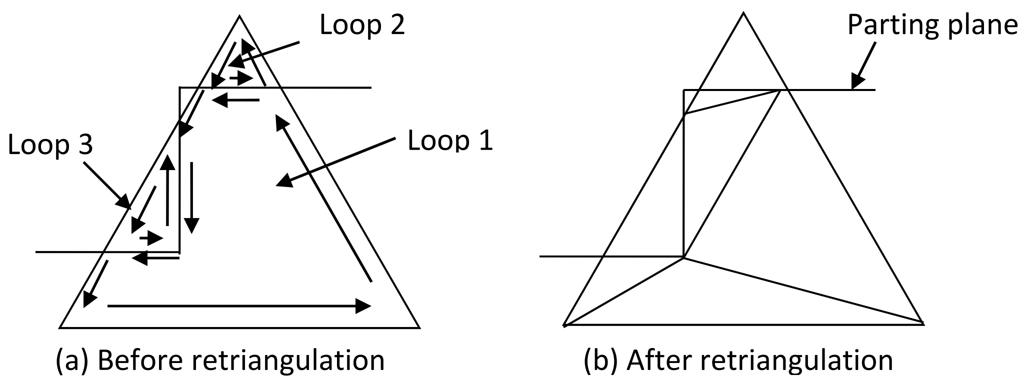


Figure 2.56 Retriangulation of triangles intersected by stepped parting plane

In order to have the polygon triangulation function performed correctly, the most critical thing is to sort all loops accurately (Figure 2.56). Details of the loop sorting algorithm are discussed below.

For recording edge information, an edge data structure is used. Each edge list is built with edge elements. Each edge element contains five parameters as shown in Figure 2.57.

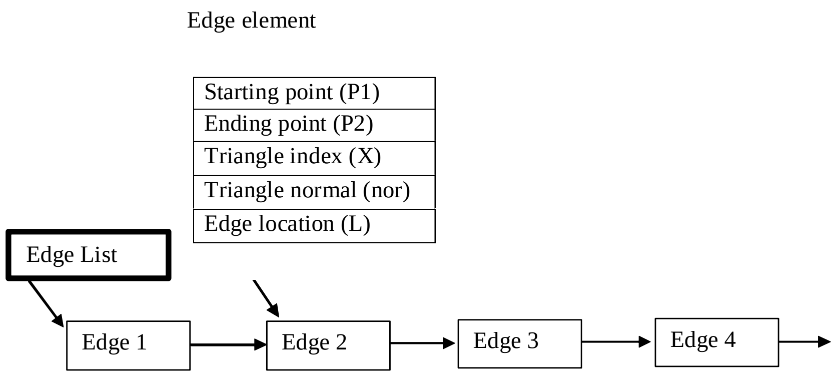


Figure 2.57 Edge data structure of retriangulated triangles

The edge location parameter L is used to record the location of the edge. If L equals -1, it means that this edge is below the parting surface and belongs to the ejector die. If L equals 1, it is above the parting surface and should be grouped into the cover die. If L equals 0, means it is not defined. This information is helpful for deciding to which die retriangulated triangles belong.

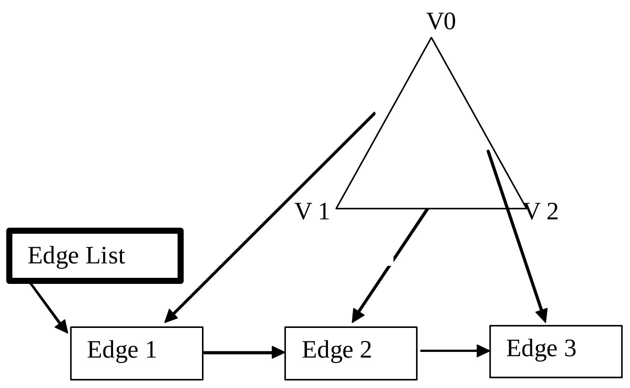


Figure 2.58 Initiation of edge list

First, for each triangle that needs to be retriangulated, a new edge list "Edge_List" is initialized with the three edges of the original triangle, and each edge's location parameter is set to zero (Figure 2.58). Second, actual values of the edge location of the triangle's three edges must be assigned. Since there is the possibility that a single triangle intersects with multiple parting surfaces, one edge might be below one parting surface but above or intersecting with another parting surface. So the edge location can not be determined by comparing its coordinate with a single parting surface. Each individual vertex position according to the parting surface to which the vertex belongs, must be calculated first (Fig. 4.21).

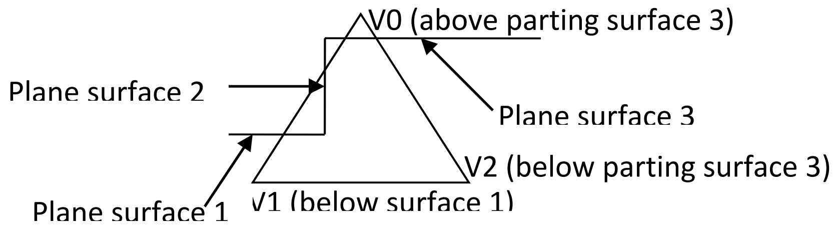


Figure 2.59 Identification of vertex location

By projecting each vertex to every plane surface and comparing the boundary of each plane surface with the coordinates of each vertex, the so called associated plane surface of the vertex can be defined if the vertex is projected inside the plane surface boundary. For example, projecting v1 to plane surface 1, 2 and 3, it is inside the surface 1's boundary but outside of surface 2 and 3, so its associated surface is plane surface 1 and its location value is recorded as -1. By applying the same algorithm to vertex 0 and 2, their location values should be 1 and -1 consecutively.

The next step is to check if both vertices of each edge have the same location value. If both location values equal to 1, the edge's location value should be 1. If both location values equal to -1, the edge's location equals to -1. Otherwise, its location value should be zero. It means this edge crosses at least one of the plane surfaces and needs to be split into multiple edges.

In order to find the new edges that created by intersection between triangle edges and at least one plane surface, the following splitting method is applied. Suppose there are N plane surfaces in a stepped parting plane. The intersection calculation is decomposed into N steps. In each step, only a single plane surface intersects with the triangle (Figure 2.60).

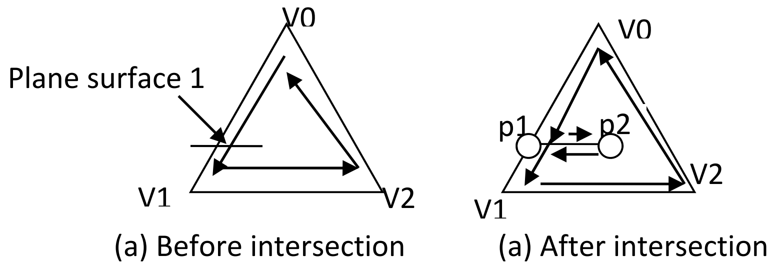


Figure 2.60 Edges resulted from intersecting triangle with first parting plane surface

As shown in Figure 2.60, edge (p1, p2) is one element of the contour list calculated when the die configuration was defined, so it is already known. Therefore, new edge (p1, p2) and edge (p2, p1) can be directly pushed into the Edge_List. The next step is to check if point p1 or p2 is on any of the triangles. If P1 is on edge (V0, V1), it means that edge (V0,V1) no longer exists and P1 split edge 1 into two edges: edge (V0, P1) and edge (P1, V1). In this case, assign point P1 to the second point of edge 1 and it changes to edge (V0, P1). In addition, a new edge (P1, V1) is inserted into the edge list too (Figure 2.61).

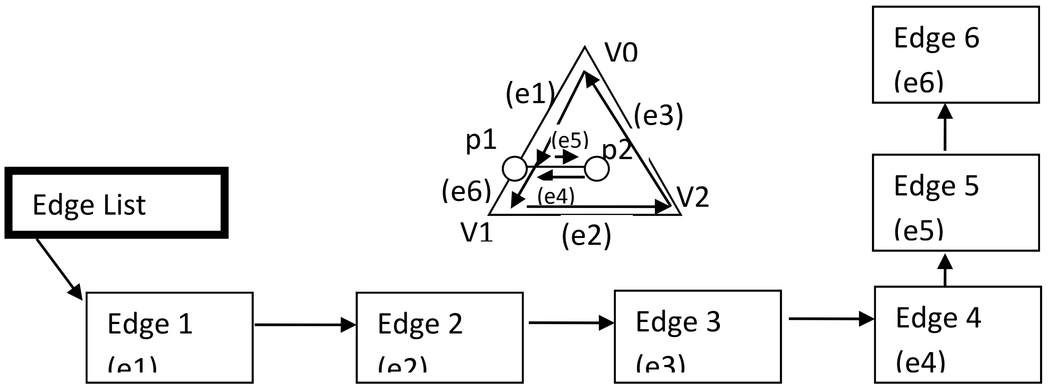


Figure 2.61 Data structure of edge list after triangle intersecting with a single plane surface

Repeat the same calculation to all the other plane surfaces that intersect with this triangle. The result should be a list that contains all the edges as shown in Figure 2.62.

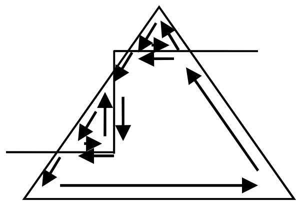


Figure 2.62 Result of edge list

At this point all the edge elements' location are arbitrary and need to be sorted. The sorting algorithm is described as in Figure 2.63.

```
1. Create a new empty edge list New_List;
2. Edge1= First edge in the original edge list;
3. Push Edge 1 into New_List and delete it from old list;
4. For(i=0; i<size of edge list L1; i++)
       ▪ Temp edge=edge i;
       ▪ If (end point of Edge 1 =starting point of temp edge)
         {
             • Edge1=edge i;
             • Jump to step 3;
         }
```

Figure 2.63 Algorithm for organizing edge list

Figure 2.64shows the final result after sorting out the loops. As long as all the loops are ready, polygon triangulation is applied to triangulate.

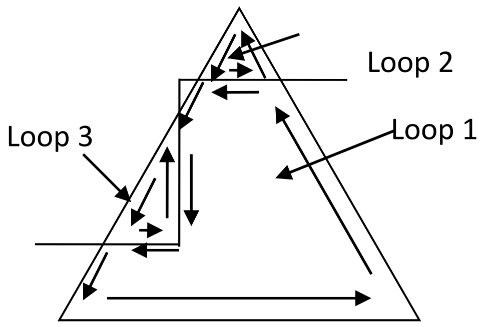


Figure 2.64 Organized triangle loops created by stepped parting plane

Figure 2.65 and Figure 2.66 are examples of die models created after triangulation and retriangulation.

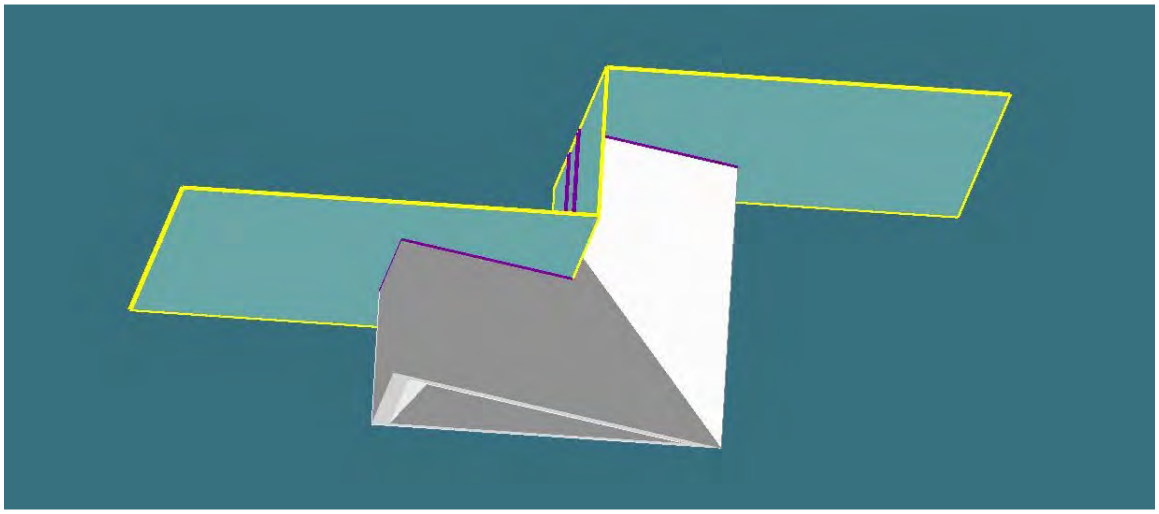
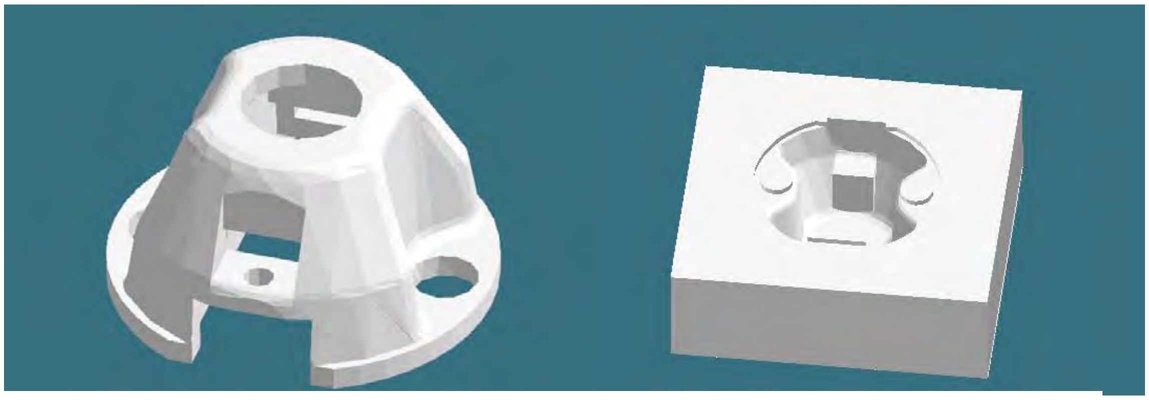


Figure 2.65 Ejector die model with stepped parting plane



(a) Part model (b) Cover die model

Figure 2.66 Cover die model for a complex shaped part

2.6 Die Property Calculations

2.6.1 Introduction

In previous sections, a methodology for constructing geometrical models of dies was discussed. With the input of the die open direction and parting plane, holes and undercuts can be automatically detected and STL models of the die components can be automatically generated. The STL models can be imported into other application such as gating system design, thermal analysis and prototyping. Furthermore, the modeling results can also provide quantitative information for further analysis, such as clamping force, machine selection, etc. The calculation of projected area, surface areas and volumes of part and dies will now be presented by utilizing the same modeling results.

2.6.2 Part Projected Area Calculation

Projected area is an important factor in die casting machine selection process. The projected area is a key factor determining the clamping force requirement [21].

The part projected area can be calculated easily because of the triangle classification. As discussed previously, triangles of the part surface are all grouped into 3 categories, 1) cover die triangle, 2) ejector die triangle, 3) undercut triangle. The projected area is the sum of ejector die triangles' projected areas (Figure 2.67).

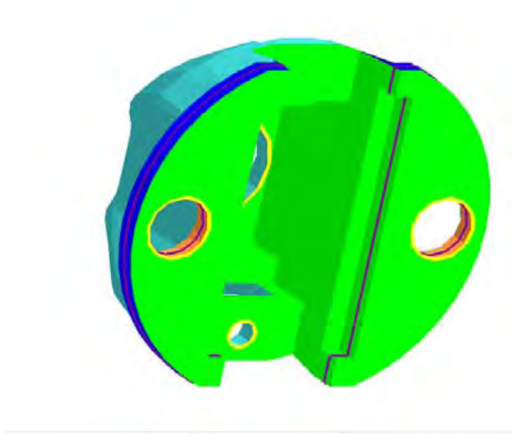


Figure 2.67 Display of ejector die triangles in green

A single triangle's projected area is computed as following:

A. Project a non-vertical single triangle to XOY plane, and a new triangle is created. Each vertex of the new triangle has the same x and y coordinates as the original vertex, but with a z coordinates equal to zero. The projected area is the area of the new triangle (Fig. 5.2).

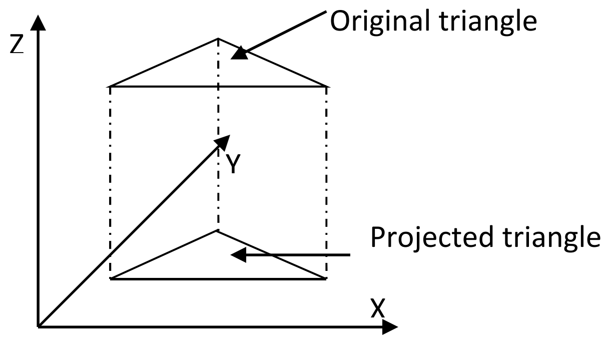


Figure 2.68 Projected area of a triangle

B. Calculate the new triangle's area. A single triangle's area is calculated by using the following equation (a, b and c are the length of each edge):

$$Area = \sqrt{s(s-a)(s-b)(s-c)} \qquad (1)$$

$$s = \frac{a+b+c}{2}$$

$$\text{Projected Area} = \sum_{i=1}^{i=n} \text{ejector triangle[i] projected area} \qquad (2)$$

The total projected area can be expressed as equation (2) (n is the total number of ejector triangles:

2.6.3 Part and Die Cavity Surface Area Calculation

Surface area is the summation of the area of surface triangles. A single triangle's surface area is also calculated by using equation (1).

Since there are triangle surface model for the part, the cover die and the ejector die, the surface area calculations are very simple. For each model, the surface area is calculated by adding up the areas of triangles belonging to that model.

2.6.4 Volume Calculation

Because each dexel is just a line segment, it is hard to use line segments display a 3D solid model. Therefore, cuboids were built for display purpose. Each cuboid has a dexel line as its centerline and square cross section with a length of the dexel line. The volume of each cuboid is the multiplication of the dexel line’s length and the area of cross section. Taking this one-step further, the volume of part, cover die, ejector die and undercut are the summation of the volumes of dexels of the respective types (Figure 2.69).

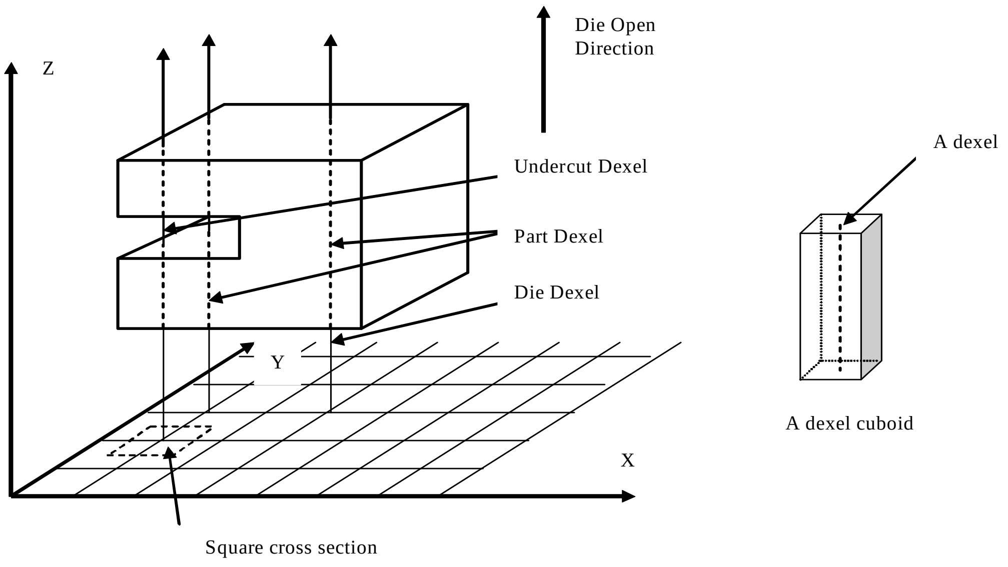


Figure 2.69 Volume calculation [21]

2.7 Conclusions

A geometric based die modeling method was proposed and implemented. This tool can be very helpful for engineers to visualize their design ideas, compare and evaluate different designs very efficiently.

Contributions of this research can be summarized as following:

- A new methodology was proposed for building a geometric based die design tool. The methodology was then implemented to build a prototype software tool, which can be used to explore, visualize, and evaluate different die design plans quickly and easily.
- By using this tool, new die models can be generated in the form of a STL data structure and presented in 3D computer graphics. Cover die and ejector die models are automatically generated using the input of die opening direction and parting plane surfaces. Undercut and holes, if any, are also automatically identified.

2.8 References

1. K. S. Lee, C. luo. (2002). Application of Case-Based reasoning in Die-Casting Die Design. Int J Adv Manuf Technol 20: 284–295
2. Shehata, F., & Abd-Elhamid, M. (2003). Computer aided foundry die-design. *Materials and Design*, *24*(8), 577-583.

3. Weishan, Z., Shoumei, X., & Baicheng, L. (1997). Study on a CAD/CAE System of Die Casting. *Journal of Materials Processing Technology*, *63*(1-3), 707-711.
4. Chen, Xiaorui (2002). Graphical user interface for cooling line functions and surface rendering.*Thesis for Master's degree, The Ohio State University*.
5. Wang, Dongtao (2004). Equilibrium temperature analysis and fill pattern reasoning for die casting process. *Ph.D. Dissertation, The Ohio State University.*
6. Y.K Woon, K. S. Lee (2004). Development of a die design system for die casting. *Int J Adv Manuf Technol 23: 399–411*
7. Wu, S. H., Lee, K. S., &Fuh, J. Y. H. (2002). Feature-Based Parametric Design of a Gating System for a Die-Casting Die. *International Journal of Advanced Manufacturing Technology*, *19*(11), 821 - 829.
8. Roller, D. (1991). An approach to computer-aided parametric design. *Computer Aided Design*, *23*(5), 385-391
9. D. Roller (1991). An approach to computer-aided parametric design. *Computer-Aided Design, 23(5), pp. 385–391*.
10. J. Y. Lee and K. Kim (1996). Geometric reasoning for knowledge-based parametric design using graph representation. *Computer-Aided Design, 28(10), pp. 831–841.*
11. Choi JC, Kwon TH, Park JH, Kim JH, Kim CH (2002). A study on development of a die design system for die-casting. *Int J Adv Manuf Technol 20:1–8*
12. Y. M. Chen and C. L. Wei (1997). Computer-aided feature-based design for net shape manufacturing. *Computer Integrated Manufacturing Systems, 10(2), pp. 147–164.*
13. A. Voss (1997). Case reusing systems – survey, framework and guidelines. *Knowledge Engineering Review, 12(1), pp. 59–89.*
14. C. K. Kwong, G. F. Smith and W. S. Lau. (1997). Application of case based reasoning in injection moulding. *Journal of Materials Processing Technology, 63, pp. 463–467.*
15. K. Hua and B. Faltings (1993). Exploring case-based design – CADRE. Artificial Intelligence in Engineering Design, Analysis and Manufacturing, 7, pp. 135–144.
16. I. Smith, C. Lottaz and B. Faltings (1995). Spatial composition using cases: IDIOM. Proceedings of First International Conference on Case-Based Reasoning", pp. 88–97, Springer, Berlin.
17. S. H. Sun and J. L. Chen (1996). A fixture design system using casebased Reasoning. *Engineering Applications of Artificial Intelligence, 9(5), pp. 533–540.*
18. B. Humm, Ch. Schulz, M. Radtke and G. Warnecke (1991). A system for case-based process planning. *Computers in Industry, 17, pp. 169–180.*
19. A. Kaufman, K.H. Hohne, W. Kruger and L. Rosehblum (1994). Research Issues in Volume Visulizaiton. *Visualization Report, pp 63-67.*
20. A. Kaufman, D. Cohen, and R. Yagel (1993). Volume Graphics. *Computer, Vol. 26, No. 7, July 1993, pp. 51-64.*

21. L. Cai (2002). Dexel-based Geometric reasoning and visualization for die configuration. *PhD dissertation, The Ohio State University.*
22. Van Hook T (1986). Real Time Verification of Multi-axis NC Programs with Raster Graphics. IEEE Proc. Of 1986 Int'l Conf. on Robotics and Automation, San Francisco. Apr. 7-10. pp.166-171.